08:36:59 OCA PAD AMENDMENT - PROJECT HEADER INFORMATION 08/12/91

Active

Project #: C-36-636 Cost share #: C-36-383 Rev #: 4
Center # : 10/24-6-R7035-0A0 Center shr #: 10/22-1-F7035-0A0 OCA file #:
Work type : RES
Contract#: CCR-9011981 Mod #: AMENDMENT 02 Document : GRANT
Prime #: Contract entity: GTRC

Subprojects ? : N CFDA: 47.070
Main project #: PE #: N/A

Project unit: COMPUTING Unit code: 02.010.300
Project director(s):
AKYILDIZ I F COMPUTING (404)894-5141

Sponsor/division names: NATL SCIENCE FOUNDATION / GENERAL
Sponsor/division codes: 107 / 000

Award period: 900801 to 930131 (performance) 930430 (reports)

Sponsor amount	New this change	Total to date
Contract value	0.00	180,690.00
Funded	85,061.00	168,084.00
Cost sharing amount		8,847.00

Does subcontracting plan apply ?: N

Title: MULTI JOB CLASS QUEUEING NETWORKS WITH BLOCKING

PROJECT ADMINISTRATION DATA

OCA contact: Mildred S. Heyser 894-4820

Sponsor technical contact

YECHEZKEL ZALCSTEIN
(202)357-7349

NSF
1800 G STREET, N.W.
WASHINGTON, D.C. 20550

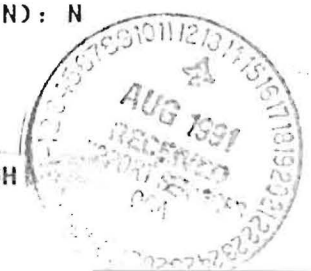
Sponsor issuing office

SHERRYE L. MCGREGOR
(202)357-7544

NATIONAL SCIENCE FOUNDATION
1800 G STREET, N.W.
WASHINGTON, D.C. 20550

Security class (U,C,S,TS) : U ONR resident rep. is ACO (Y/N): N
Defense priority rating : N/A X supplemental sheet
Equipment title vests with: Sponsor GIT X

Administrative comments -
AMENDMENT 2 ADDS $85,061 AND EXTENDS PERFORMANCE ENDING DATE THROUGH JANUARY 31, 1993.



GEORGIA INSTITUTE OF TECHNOLOGY
OFFICE OF CONTRACT ADMINISTRATION

NOTICE OF PROJECT CLOSEOUT

Closeout Notice Date 05/24/93

Project No. C-36-636 Center No. 10/24-6-R7035-0A0

Project Director AKYILDIZ I F School/Lab COMPUTING

Sponsor NATL SCIENCE FOUNDATION/GENERAL

Contract/Grant No. CCR-9011981 Contract Entity GTRC

Prime Contract No. _____

Title MULTI JOB CLASS QUEUEING NETWORKS WITH BLOCKING

Effective Completion Date 930131 (Performance) 930430 (Reports)

Closeout Actions Required:	Y/N	Date Submitted
Final Invoice or Copy of Final Invoice	Y	
Final Report of Inventions and/or Subcontracts	N	
Government Property Inventory & Related Certificate	N	
Classified Material Certificate	N	
Release and Assignment	N	
Other _____	N	

CommentsEFFECTIVE DATE 8-1-90. CONTRACT VALUE $180,690.

Subproject Under Main Project No. _____

Continues Project No. _____

Distribution Required:

Project Director	Y
Administrative Network Representative	Y
GTRI Accounting/Grants and Contracts	Y
Procurement/Supply Services	Y
Research Property Managment	Y
Research Security Services	N
Reports Coordinator (OCA)	Y
GTRC	Y
Project File	Y
Other CARL BAXTER-FMD	Y
FRED CAIN-OOD	Y

ANNUAL PROGRESS REPORT FOR CONTRACT No. CCR-90-11981

MULTI JOB CLASS QUEUEING NETWORKS WITH BLOCKING

Ian F. Akyildiz

College of Computing

Georgia Institute of Technology

Atlanta, GA 30332

U.S.A.

May 29, 1991

Deliverable No. 1

Period: August 1, 1990 - July 31, 1991

1 Scientific Results

The overall scientific objective of this research is to analyze queueing networks with blocking. These are networks with finite buffers where the blocking occurs when the flow of jobs through one station may be stopped for a moment if a destination station has reached its full capacity. One key aspect of our work is focused on investigating the deadlock freedom of these networks. Another aspect is to develop exact as well as approximate solution techniques for the analysis of these networks. Another goal is to find applications of these models and show that our results provide accurate performance results. Significant progress has been made along each of these fronts.

2 Already Achieved Goals

2.1 Deadlock Freedom

Blocking can lead to a deadlock state due to the finite buffer capacities in queueing network models. In [1,2] we investigated the deadlock freedom in multiple chain queueing networks with blocking, and gave necessary and sufficient conditions for deadlock freedom. We presented an algorithm to find a deadlock free buffer allocation with the least number of buffers, thus the optimal allocation. The algorithm maps the queueing network into a directed graph. A procedure for finding a subset of all cycles in the directed graph is used to generate the conditions for deadlock freedom of the network. Finally, we obtained the optimal buffer allocation from these conditions by applying linear programming techniques.

2.2 Application on ATM Switches

ATM (Asynchronous Transfer Mode) is recommended as the basis for the future broadband ISDN. One of critical issue in designing and managing an ATM network is how to serve the network traffics of different types differently so as to meet each of their service requirements without the use of the unnecessarily large network capacity. In [2] we analyze an $M_1, M_2/G_1, G_2/N$ push-out queueing model, which could be used to evaluate the performance of an output link of ATM switches with two-class priority traffics. Our model differs from the previous work in the following. General service time distributions, which could be different for classes 1 and 2, are allowed. A general

service discipline function, $\alpha_1(i,j)$, is introduced where $\alpha_1(i,j)$ is the probability with which the next service is for a class 1 packet, given that there are i and j class 1 and class 2 packets at the beginning of the service time. Many interesting service disciplines can be modeled with $\alpha_1(i,j)$, and therefore, can be analyzed with the exactly same formula. Total buffer space N is divided as $N = N_1 + N_2$. An arrival of class 2 can take any unoccupied buffer space in the system upon its arrival. An arrival of class 1 can take only an unoccupied buffer space in N_1 section. However it can push out a class 2 packet occupying a buffer space in N_1 section and not being in service if there is no unoccupied buffer space in N_1 section upon its arrival. All these mechanisms can be used individually or together to control the grade of service of an ATM network.

We develop a solution by solving a set of $N_1(2N - N_1 + 1)/2$ linear equations. We obtain exact values of loss probabilities for classes 1 and 2, the queue length distribution and the mean waiting time for class 1 and an approximate calculation for queue length distribution and mean waiting time for class 2. Numerical examples are also given in the paper.

2.3 Exact Solutions

In this part we have two major solutions. The first part deals with two-station networks with BAS mechanism and different station types. In this part only single class of jobs is allowed. The contribution here is to show that exact solutions exist for two station queueing networks with BAS mechanism having different station types. The exact equilibrium state probability distributions are derived. Insensitivity is investigated and and formulas for performance measures are obtained. It is demonstrated that the throughputs, mean number of jobs and mean number of blocked jobs depend on the scheduling discipline. A queueing network model with more than two stations is analyzed in the second part. Multiple job classes with job class change, and different station types are allowed in the model. Exact solutions for equilibrium state probabilities and performance measures are obtained under the condition that only a certain total number of jobs is allowed in the network.

We analyze two station queueing networks with different scheduling disciplines and blocking. We also analyze these queueing networks with more than two stations which may have exact solutions under the condition that the total number of jobs has a certain value. Using the equivalence between the blocking network and nonblocking network we show how to use algorithms for classical networks to compute performance measures for the queueing network with blocking.

3 Work in Progress

Currently we are working on several problems pointed out in the proposal. At the end of the summer 1991 we are planning to achieve the following immediate goals.

3.1 Mean Value Analysis for Blocking Networks

As pointed out in the proposal section 3.3 we want to find an efficient computational algorithm based on mean valuea analysis to obtain performance measures for networks of queues with blocking. This is an approxiamtion algorithm. We implemented the algorithm as suggested in the proposal and simulated several test models. Currently we are validating the approximation technique.

3.2 Open Queueing Networks with Blocking

In section 3.6 we proposed to analyze open queueing networks with blocking by showing an exact equivalency between open and closed queueing networks with blocking. Currently we are working on the proofs.

3.3 Equivalencies between Different Types of Blocking Mechanisms

In section 3.7.3 of the proposal we mention that an extensive study was carried out in the past regarding to establish the equivalencies between different types of blocking mechanisms. However, all those studies were based on exponential assumptions, single job class and basically FCFS scheduling discipline. Now we are in the process of investigating these equivalencies in the framework of our model where we have muliple job classes, different station types, and nonexponential assumptions.

Publications

1. I. F. Akyildiz and J. Liebeherr, "Optimal Deadlock Free Buffer Allocation in Multiple Chain Blocking Networks of Queues", to appear in *Proc. of the International Conference on the Performance of Distributed Systems and Integrated Communication Networks*, Kyoto, Japan, September 10-12, 1991. (Accepted in May 1991). (*This paper is a shorter version of [2]*).

2. J. Liebeherr and I. F. Akyildiz, "Deadlock Freedom in Queueing Networks with Multiple Routing Chains and Finite Capacities" *Technical Report, College of Computing, Georgia Tech, TR-CC-91-05, January 1991.* Submitted for publication to *Journal of the ACM*, March 1991.

3. I. F. Akyildiz and X. Cheng, "Performance Analysis of an ATM Switch with Different Scheduling and Push-Out Scheme", *Technical Report, College of Computing, Georgia Tech, TR-CC-91-06, January 1991.* Submitted for publication to *IEEE Transactions on Communications* Journal, January 1991.

4. I. F. Akyildiz and H. von Brand, "Exact Solutions for Multi Job Class Networks of Queues with Blocking-after-Service", *Technical Report, College of Computing, Georgia Tech, TR-CC-91-27, April 1991.* Submitted for publication to *ACTA INFORMATICA* Journal, April 1991.

FROM I. F. AKYILDIZ

ATTENTION:
TRANSFER INFORMATION FROM BLOCK 1, PART 1 HERE

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 12806 WASHINGTON, D.C.

POSTAGE WILL BE PAID BY NATIONAL SCIENCE FOUNDATION

NATIONAL SCIENCE FOUNDATION
1800 G STREET, N.W.
WASHINGTON, D.C. 20277-2806

NATIONAL SCIENCE FOUNDATION
FINAL PROJECT REPORT
ENCLOSE FORM 98A

OMB Number 3145-0058

NATIONAL SCIENCE FOUNDATION
1800 G STREET, NW
WASHINGTON, DC 20550

PI/PD Name and Address

Ian F. Akyildiz
Information and Computer Science
GA Tech Res Corp - GIT
Atlanta GA 30332

NATIONAL SCIENCE FOUNDATION
FINAL PROJECT REPORT

PART I - PROJECT IDENTIFICATION INFORMATION

1. Program Official/Org. Yechezkel Zalcstein - CCR

2. Program Name COMPUTER SYSTEMS ARCHITECTURE PROGRAM

3. Award Dates (MM/YY) From: 08/90 To: 01/93

4. Institution and Address
GA Tech Res Corp - GIT
Administration Building
Atlanta GA 30332

5. Award Number 9011981

6. Project Title
Multi-Job Class Queueing Networks With Blocking

This Packet Contains
NSF Form 98A
And 1 Return Envelope

F Grant Conditions (Article 17, GC-1, and Article 9, FDP-II) require submission of a Final Project port (NSF Form 98A) to the NSF program officer no later than 90 days after the expiration of the ard. Final Project Reports for expired awards must be received before new awards can be made SF Grant Policy Manual Section 677).

ow, or on a separate page attached to this form, provide a summary of the completed project and technical information. Be to include your name and award number on each separate page. See below for more instructions.

PART II - SUMMARY OF COMPLETED PROJECT (for public use)

summary (about 200 words) must be self-contained and intelligible to a scientifically literate reader. Without restating the ject title, it should begin with a topic sentence stating the project's major thesis. The summary should include, if pertinent he project being described, the following items:

he primary objectives and scope of the project
he techniques or approaches used only to the degree necessary for comprehension
he findings and implications stated as concisely and informatively as possible

= SEE ATTACHED =

PART III - TECHNICAL INFORMATION (for program management use)

references to publications resulting from this award and briefly describe primary data, samples, physical collections, ntions, software, etc. created or gathered in the course of the research and, if appropriate, how they are being made available e research community. Provide the NSF invention Disclosure number for any invention.

= SEE ATTACHED =

I.F. AKYILDIZ	3/3/1993
Principal Investigator/Project Director Signature	Date

IMPORTANT:
MAILING INSTRUCTIONS
Return this *entire* packet plus all attachments in the envelope attached to the back of this form. Please copy the information from Part I, Block I to the *Attention block* on the envelope.

PART IV — FINAL PROJECT REPORT — SUMMARY DATA ON PROJECT PERSONNEL

(To be submitted to cognizant Program Officer upon completion of project)

The data requested below are important for the development of a statistical profile on the personnel supported by Federal grants. The information on this part is solicited in response to Public Law 99-383 and 42 USC 1885C. All information provided will be treated as confidential and will be safeguarded in accordance with the provisions of the Privacy Act of 1974. You should submit a single copy of this part with each final project report. However, submission of the requested information is not mandatory and is not a precondition of future award(s). Check the "Decline to Provide Information" box below if you do not wish to provide the information.

Please enter the numbers of individuals supported under this grant.
Do not enter information for individuals working less than 40 hours in any calendar year.

	Senior Staff		Post-Doctorals		Graduate Students		Under-Graduates		Other Participants[1]	
	Male	Fem.	Male	Fem.	Male	Fem.	Male	Fem.	Male	Fem.
A. Total, U.S. Citizens	X									
B. Total, Permanent Residents										
U.S. Citizens or Permanent Residents[2]:										
American Indian or Alaskan Native										
Asian										
Black, Not of Hispanic Origin										
Hispanic										
Pacific Islander										
White, Not of Hispanic Origin										
C. Total, Other Non-U.S. Citizens										
Specify Country 1. FRANCE			X							
2. GERMANY					X					
3. TAIWAN					X					
D. Total, All participants (A + B + C)	1		1		2					
Disabled[3]										

☐ Decline to Provide Information: Check box if you do not wish to provide this information (you are still required to return this page along with Parts I-III).

[1] Category includes, for example, college and precollege teachers, conference and workshop participants.

[2] Use the category that best describes the ethnic/racial status for all U.S. Citizens and Non-citizens with Permanent Residency. *(If more than one category applies, use the one category that most closely reflects the person's recognition in the community.)*

[3] A person having a physical or mental impairment that substantially limits one or more major life activities; who has a record of such impairment; or who is regarded as having such impairment. *(Disabled individuals also should be counted under the appropriate ethnic/racial group unless they are classified as "Other Non-U.S. Citizens.")*

AMERICAN INDIAN OR ALASKAN NATIVE: A person having origins in any of the original peoples of North America, and who maintain cultural identification through tribal affiliation or community recognition.

ASIAN: A person having origins in any of the original peoples of East Asia, Southeast Asia and the Indian subcontinent. This area includes, for example, China, India, Indonesia, Japan, Korea and Vietnam.

BLACK, NOT OF HISPANIC ORIGIN: A person having origins in any of the black racial groups of Africa.

HISPANIC: A person of Mexican, Puerto Rican, Cuban, Central or South American or other Spanish culture or origin, regardless of race.

PACIFIC ISLANDER: A person having origins in any of the original peoples of Hawaii; the U.S. Pacific Territories of Guam, American Samoa, or the Northern Marianas; the U.S. Trust Territory of Palau; the islands of Micronesia or Melanesia; or the Philippines.

WHITE, NOT OF HISPANIC ORIGIN: A person having origins in any of the original peoples of Europe, North Africa, or the Middle East.

PART II. PRIMARY OBJECTIVES AND SCOPE OF THE PROJECT

Performance is a major issue in the design and implementation of systems such as computer systems, production systems, communication networks, and flexible manufacturing systems. The success or failures of such systems is judged by the degree to which performance objectives are met. Thus, tools techniques for predicting performance measures are of great interest. In the last two decades it has been demonstrated several times that performance can be evaluated or predicted well by queueing network models. For infinite capacity queueing networks numerous methods have been introduced in the last two decades. However, in actual systems stations (resources with buffers) have a finite capacity, queueing networks with blocking should be used for accurate modeling for performance analysis and prediction. Most of the previous studies on blocking queueing networks were restricted to exponential service time distributions, single job classes and FCFS scheduling disciplines. In actual systems the devices may have different scheduling disciplines, the job behavior may vary and the service times are drawn non-exponentially. In this project we tried to remove these limitations and found several new solutions for blocking queueing networks with different station types and multiple job classes. We believe that our solutions discovered within this project significantly advanced the queueing network theory.

• TECHNIQUES AND APPROACHES

- A queueing network is mapped to a directed graph.
- An efficient procedure is developed for finding cycles in directed graphs.
- The optimal buffer allocation problem is stated as an optimization problem.
- The optimization problem is solved using linear programming techniques.
- Several novel product form (separability) solutions are found for complex queueing network models. Product form solutions help to separate stations from each other and analyze them individually as if they were single stations.
- A complex queueing system with interesting scheduling disciplines (push-out schemes) and finite buffers are solved. The queueing system has an application in so-called B-ISDN (ATM) networks. Global-balance-equation technique is used to compute equilibrium state probabilities.

• FINDINGS AND IMPLICATIONS

- Conditions are found for deadlock-freedom in multiple chain queueing networks with blocking. This result will help to designers to determine whether their system may have deadlocks or not and also determine the opitmal bufgfer sizes in their system such that no deadlocks will occur. This issue is very important in designing of computer architecture and telecommunication and manufacturing systems.

- The queueing model solved in [5,7] will help to design an efficient output buffer for ATM switches in B-ISDN networks.

- Product form solutions found in all remaining papers will help to carry out efficient performance analysis of systems from computer arcitecture, telecommunication, and manufacturing areas. The capabilities of performance analysis methods have been expanded in this project.

PART III. TECHNICAL INFORMATION

- **Deadlock Freedom.**

Blocking can lead to a deadlock state due to the finite buffer capacities in queueing network models. In [3,9,11] we investigated the deadlock freedom in multiple chain queueing networks with blocking, and gave necessary and sufficient conditions for deadlock freedom. We presented an algorithm to find a deadlock free buffer allocation with the least number of buffers, thus the optimal allocation. The algorithm maps the queueing network into a directed graph. A procedure for finding a subset of all cycles in the directed graph is used to generate the conditions for deadlock freedom of the network. Finally, we obtained the optimal buffer allocation from these conditions by applying linear programming techniques.

- **Application to ATM Switches.**

ATM (Asynchronous Transfer Mode) is recommended as the basis for the future broadband ISDN. One of the critical issues in designing and managing an ATM network is how to serve the network traffics of different types differently so as to meet each of their service requirements without the use of the unnecessarily large network capacity. In [5,7] we analyze an $M_1,M_2/G_1,G_2/N$ push-out queueing model, which could be used to evaluate the performance of an output link of ATM switches with two-class priority traffics. We obtain exact values of loss probabilities for classes 1 and 2 due ti finite buffer, the queue length distribution and the mean waiting time for class 1 and an approximate calculation for queue length distribution and mean waiting time for class 2. Numerical examples are also given in the paper.

- **Exact Solutions for Networks with Blocking-after-Service**

In [6] we have two major solutions. The first part deals with two-station networks with BAS mechanism and different station types. In this part we allow only single class of jobs. The contribution here is to show that exact solutions exist for two station queueing networks with BAS mechanism having different station types. We derive exact equilibrium state probability distributions. We investigate the Insensitivity property. We demonstrate that the throughputs, mean number of jobs and mean number of blocked jobs depend on the scheduling discipline. In the second part of [6] we analyze a queueing network model with more than two stations. We have multiple job classes with job class change, and different station types in the model. We obtained exact solutions for equilibrium state probabilities and performance measures under the condition that only a certain total number of jobs is allowed in the network.

- **Exact Analysis of General Topology Queueing Networks with Blocking-After-Service Mechanism**

In [4,10] we analyze two models of closed queueing networks with blocking-after-service and multiple job classes. The first model is a network with N stations and each station has either Type *II* or Type *III*. The second model is a star like queueing network also called as central server model in which the stations may have either Type *I* or Type *IV*, with the condition that the neighbors of these stations must be of Type *II* or Type *III* such that blocking will be caused only by this set of station types. We obtained exact product form solutions for the equilibrium state probabilities in both models. We also derived formulae for performance measures such as throughput and mean number of jobs.

- **Neo-Queueing Networks with Blocking.**

In [1] we analyzed three different queueing network models with infinite and finite buffer capacities. In the first model the transition of jobs is different than in the classical queueing network models. A job completing its service and being routed to a destination station will choose its next destination station before it enters the first destination station. The routing probabilites are then specified accordingly. In the finite buffer case blocking occurs before the job enters the server and it checks the next destination station whether its buffer capacity is full or not. If the buffer is full, the job is forced to stay at the head of the queue and blocks the server to serve other jobs. In the second model whenever a job gets its service there is also a time limit for that job to

complete its service. If the service cannot be completed before the limited time, then the job is forced to leave and is routed to another station by different routing probability than the routing probabilities after normal service. In the finite buffer case blocking occurs after the service. where a job completing its service at a station cannot proceed to next station due its full buffer capacity. The job is forced to wait in the server of the station until a place will become available in the destination station. The third model contains single negative job class and multiple positive job classes. When a negative job arrives at a station one of the positive jobs, if there is any, will be killed by this negative job. For Type IV stations the job in service will be killed where for Type II stations any job has the same probability to be killed. Each station has a finite buffer capacity and jobs can be blocked due to so-called repetitive-service-blocking. We have shown that all models have an exact product form solution.

- **Kanban Blocking.**

In [2] We obtain an exact product form solution for a queueing network with Type I, II, III and IV stations having finite buffer capacities. A job is blocked after service if its destination is full but the server continues to serve the unblocked jobs in its station. This type of blocking is called Kanban Blocking. The service time distributions are assumed to be exponential for Type I and Coxian for Type II, III, and IV. It is also assumed that in the beginning of the service each phase of the Coxian representation may be reached with a positive probability. The steady-state distribution of the number of blocked and unblocked jobs depends on the rates of the Cox phases. Thus, the insensitivity property does not hold in this model in contrast to the classical BCMP queueing networks. We developed a convolution algorithm for the computation of the normalization constant. We derived new formulae for the computation of performance measures.

- **Different Blocking Types.**

In [8] it is proven that a multi-job class open queueing network with finite capacity queues is *pathwise-equivalent* to a closed queueing network model with finite buffers. This implies particularly that the two queueing networks have the same queue length distribution. Furthermore, the service and interarrival time distributions are assumed to be general. It is proven that such networks are irreducible if the underlying networks with infinite capacities are irreducible. Finally, the steady-state distribution of a multi-job class queueing network with finite buffers is computed using the steady-state distribution of a single job class queueing network with finite buffers. Moreover, performance measures are computed for the individual classes using only the global performance measures of the constructed single class queueing network.

REFEREED PUBLICATIONS

I. REFEREED JOURNAL PUBLICATIONS

SUBMITTED.

1. I. F. Akyildiz and E. Gelenbe, *Product Form Results for Novel Queueing Networks*. To be submitted.

2. I. F. Akyildiz, *Exact Analysis of Closed Queueing Networks with Kanban Blocking*, **Queueing Systems: Theory and Applications**, March 1993.

3. I. F. Akyildiz and J. Liebeherr, *Deadlock Freedom in Queueing Networks with Multiple Chains and Blocking*, **Performance Evaluation Journal**, December 1992.

TO APPEAR.

4. I. F. Akyildiz and C. C. Huang, *Exact Product Form Solution for Queueing Networks with Multiple Job Classes and Blocking-after-Service*. **Queueing Systems: Theory and Applications**, 1993.

5. I. F. Akyildiz and X. Chen, *A Queueing System with Finite Buffer and Different Push Out Schemes*, **Performance Evaluation Journal**, 1993.

6. I. F. Akyildiz and H. von Brand, *Exact Analysis of Networks of Queues with Blocking-After-Service*, **Theoretical Computer Science Journal**, 1993.

II. REFEREED CONFERENCE PUBLICATIONS

7. X. Chen and I. F. Akyildiz, *A Finite Buffer Two Class Queue with Different Scheduling and Push-Out Schemes*, *Proc. of the* **IEEE INFOCOM'92**, Florence, Italy, May 1992, pp. 231-242. (Preliminary version of Paper #5).

8. T. Choukri, *Exact Analysis of Multiple Job Classes and Different Types of Blocking*, **Proc. of the Second Workshop on Queueing Networks with Blocking**, North-Holland Publ. Comp., May 1992, RTP/NC, pp.15-27.

9. J. Liebeherr, *Deadlock-Free Buffer Allocations in Multiple Chain Blocking Networks with Tandem Sequences* **Proc. of the Second Workshop on Queueing Networks with Blocking**, North-Holland Publ. Comp., May 1992, RTP/NC, pp.41-55.

10. I. F. Akyildiz and C. C. Huang, *Exact Analysis of Multi-Job Class Networks of Queues with Blocking-after-Service*, **Proc. of the Second Workshop on Queueing Networks with Blocking**, North-Holland Publ. Comp., May 1992, RTP/NC, pp.143-157. (Preliminary version of Paper #4).

11. I. F. Akyildiz and J. Liebeherr, *Optimal Deadlock Free Buffer Allocation in Multiple Chain Networks of Queues with Blocking*, **Proc. of the Int. Conf. on Parallel and Distributed Systems Performance**, North-Holland, Kyoto/Japan, September 1991, pp.217-236. (Preliminary version of Paper #3).

REFEREED PUBLICATIONS

I. REFEREED JOURNAL PUBLICATIONS

SUBMITTED.

1. I. F. Akyildiz and E. Gelenbe, *Product Form Results for Novel Queueing Networks*. To be submitted.

2. I. F. Akyildiz, *Exact Analysis of Closed Queueing Networks with Kanban Blocking*, **Queueing Systems: Theory and Applications**, March 1993.

3. I. F. Akyildiz and J. Liebeherr, *Deadlock Freedom in Queueing Networks with Multiple Chains and Blocking*, **Performance Evaluation Journal**, December 1992.

TO APPEAR.

4. I. F. Akyildiz and C. C. Huang, *Exact Product Form Solution for Queueing Networks with Multiple Job Classes and Blocking-after-Service*. **Queueing Systems: Theory and Applications**, 1993.

5. I. F. Akyildiz and X. Chen, *A Queueing System with Finite Buffer and Different Push Out Schemes*, **Performance Evaluation Journal**, 1993.

6. I. F. Akyildiz and H. von Brand, *Exact Analysis of Networks of Queues with Blocking-After-Service*, **Theoretical Computer Science Journal**, 1993.

II. REFEREED CONFERENCE PUBLICATIONS

7. X. Chen and I. F. Akyildiz, *A Finite Buffer Two Class Queue with Different Scheduling and Push-Out Schemes*, *Proc. of the* **IEEE INFOCOM'92**, Florence, Italy, May 1992, pp. 231-242. (Preliminary version of Paper #5).

8. T. Choukri, *Exact Analysis of Multiple Job Classes and Different Types of Blocking*, **Proc. of the Second Workshop on Queueing Networks with Blocking**, North-Holland Publ. Comp., May 1992, RTP/NC, pp.15-27.

9. J. Liebeherr, *Deadlock-Free Buffer Allocations in Multiple Chain Blocking Networks with Tandem Sequences* **Proc. of the Second Workshop on Queueing Networks with Blocking**, North-Holland Publ. Comp., May 1992, RTP/NC, pp.41-55.

10. I. F. Akyildiz and C. C. Huang, *Exact Analysis of Multi-Job Class Networks of Queues with Blocking-after-Service*, **Proc. of the Second Workshop on Queueing Networks with Blocking**, North-Holland Publ. Comp., May 1992, RTP/NC, pp.143-157. (Preliminary version of Paper #4).

11. I. F. Akyildiz and J. Liebeherr, *Optimal Deadlock Free Buffer Allocation in Multiple Chain Networks of Queues with Blocking*, **Proc. of the Int. Conf. on Parallel and Distributed Systems Performance**, North-Holland, Kyoto/Japan, September 1991, pp.217-236. (Preliminary version of Paper #3).

Product Forms for Novel Queueing Networks with Blocking

Ian F. Akyildiz†and Erol Gelenbe‡

†School of Electrical Engineering, Georgia Institute of Technology, Atlanta, GA 30332 (USA).

‡EHEI, Universite de Paris V, 45 rue Saint Peres, 75006 Paris (France).

Abstract. Three new multiple job class queueing network models with finite buffer capacities are *shown to have product form.* In the *first model*, which has "early routing decision", a job chooses its destination and the next station after that, according to a Markovian rule as soon as it enters a station, and when it reaches the head of the queue it will block the server stopping it from serving jobs if both station's buffers are full. In the *second model*, each job has a deadline by which it must complete service. If the job is still in service when the time limit runs out, it will leave that station and is routed to another station according to a routing probability which is distinct from the one after normal service completion. Here blocking occurs after service when a job completing service cannot proceed to the next station if its buffer is full, and thus blocks its current server. The *third model* is derived from networks with "positive and negative jobs", and contains a single negative and multiple positive job classes. Each station has finite buffer capacity and jobs can be blocked with "repetitive-service-blocking" if the station they need to go to is full.

Key Words: *Performance Evaluation, Queueing Networks, Product Form Solutions, Infinite Buffer Capacity, Finite Buffer Capacity, Blocking, Impatient Jobs, Early Routing, Positive and Negative Jobs, Equilibrium State Probabilities.*

1 Introduction

Queueing network models are now well established as useful tools for the study of diverse systems such as computer-communication networks, high-speed communication networks, transportation

*This work was supported by National Science Foundation (NSF) under Grant No. CCR-90-11981 (for the first author), and by the European Economic Community via ESPRIT BRA Project QMIPS (for the second author).

networks, production management and manufacturing systems, inventory control models, computer system performance models, and even population models for different biological entities.

Over the years, even though it has often been predicted that new product form results are no longer attainable, new product form results have been found; recent work includes [2,3,7,8,11,12]. In this paper we contribute some new results in this direction. We consider queueing networks with multiple class jobs and stations with various service disciplines and finite capacity queues, in extension of the seminal work in [6].

In the basic queueing network model we consider here, there are N stations, a total of K jobs and R job classes. Each station may belong to one of the four following *Types* [6]:

- Type I: The service discipline is first-come-first-served (FCFS); all job classes have the same service time distribution and the service rate can be state-dependent where $\mu(k)$ will denote the service rate with k jobs.
- Type II: There is a single server and the service discipline is processor sharing (PS). Each job class may have a distinct service time distribution.
- Type III: The number of servers is greater or equal to the maximum number of jobs which can be queued at the station; this is known as an infinite server station (IS). Each job class may have a distinct service time distribution.
- Type IV: There is a single server and the queueing discipline is last-come-first-served preemptive-resume (LCFS-PR). Each job may have a distinct service time distribution.

In the original "BCMP" model [6], jobs move from one station to another according to a Markovian transition rule giving the quantity $p_{i,r;j,s}$, which is the probability that a class r job leaving station i after service, then enters station j as a job of class s.

In this paper we will consider three different *blocking networks* having finite capacity queues, which are extensions of BCMP networks [6] or of the *Gelenbe Networks* introduced in [8], and show that they all have *product form*. Each model will admit a specific type of blocking in the case of finite capacity, and will have characteristics due to the four Types of stations. In this section we briefly explain these three models.

The *first model* has "early routing decisions", where jobs choose their destination station and the station after that according to a Markovian rule as soon as they enter a station contrary to standard networks where the next station is chosen after the current service. Blocking occurs before the job enters the server, when it checks the destination station and the next station after that and discovers that their buffer capacities are full. If one (or both) of their buffers is full, the job is forced to stay at the head of the queue and stops the current server from serving other jobs until it can move into the next station.

The *second model* considers the case where jobs have a deadline to complete service, where this deadline is expressed as a time starting when service begins. If a job is still in service when the deadline or time limit runs out, it is forced to leave that station and is routed to another station according to a distinct routing probability than the one after normal service completion. This model can be be viewed as an instance of the Gelenbe Networks [12] with triggered job motion, in which queues have finite capacity. Here we assume that blocking occurs after service, when a job completing service cannot proceed to the next station if its buffer is full, waits at the current station and blocks its service.

The *third model* we consider is derived from networks with "positive and negative jobs" [8], containing a single negative job class and multiple positive job classes as in [7]. Each station has finite buffer capacity and jobs can be blocked with "repetitive-service-blocking" if the station they need to proceed is full.

2 Queueing Networks with Early Routing Decisions, Finite Buffer Capacity, and Extended Blocking Before Service

A queueing network will said to have *early routing decisions*, if as soon as a job arrives at a station in some class, the next station it will visit and its next class is determined via the Markov chain with transition probabilities $(p_{i,r;j,s})$ where (i,r) denotes the (station, class) pair of the job as it enters the current station, while (j,s) is the corresponding pair at the next station it will visit. Thus, in this model the network state must include information about the *next* (station, class) pair. A direct application of this model is the communication networks where an incoming packet is assigned its outgoing link upon arrival to the node. This model is, of course, motivated by networks in which "virtual circuits" are established.

Clearly, in the infinite buffer capacity case, this assumption merely leads to a special case of the BCMP theorem [6]; indeed, all information about routing (including deterministic finite routing in a finite network, which specifies *a finite sequence of all the stations* which each job will visit, may be represented by a finite number of job classes. The novelty in this section is that we consider a network with queues of *finite* capacity, which when combined with early routing decisions, leads to some interesting and useful results.

Thus in this section, we assume that the length of station i is limited to some finite B_i, $1 \leq i \leq N$. We also assume that the network is *deadlock-free* [5], i.e.,

$$K < \min_{p_l \in \mathcal{P}} \sum_{i \in P_l} B_i, \tag{1}$$

where K is the total number of jobs in the system, $\mathcal{P}$ denotes the set of all routing paths of the network, and p_l denotes one of the paths in $\mathcal{P}$.

The finite capacity of queues in a network may, in general, lead to blocking. The type of blocking we consider here is a generalization of *Blocking-Before-Service* (BBS). In conventional BBS, blocking occurs just before a job enters the server (i.e. when it is at the head of the queue, or about to begin service); at that time it checks its destination to see whether its buffer is full or not. If it is full, then the job remains at the head of the current queue and stops the server from serving other jobs.

In the case we consider in this section, which we call *E-BBS (Extended-BBS)*, the job checks its destination **and** the next station it will visit (since this is the information provided by the routing probability), and will block the current server if either one (or both) of its two stations downstream is full. Clearly, E-BBS is motivated by the "virtual circuit" type of behaviour of certain communication networks, and is a step towards modeling such behaviour. In the sequel we restrain our attention to the case of a *single job class*, and exhibit and prove the product form solution for E-BBS.

The state space of the network is $\mathcal{S} = (\mathbf{x}, \mathbf{y})$ where $\mathbf{x} = (\mathbf{x_1} \ldots, \mathbf{x_N})$, and $\mathbf{x_i} = (\mathbf{x_{i1}}, \mathbf{x_{i2}}, \ldots)$ is the vector of nonblocking jobs at station i, giving the next station to be visited by the jobs in the queue in a form appropriate for the Type of station; i.e. if the station is of Type I or Type IV, then $\mathbf{x_{i1}}$ is the first job that will be served in that order, while for the two other Types of stations the order does not matter. The vector $\mathbf{y} = (\mathbf{y_1}, \ldots, \mathbf{y_N})$, contains information on blocked jobs with $\mathbf{y_i} = (y_{i1}, y_{i2}, \ldots)$, where y_{il} denotes the location (i.e. station number) of the l-th job blocked by station i.

Let α_i denote the total arrival rate of jobs to station i; consistent with this notation, $\alpha_{y_{il}}$ denotes the arrival rate of jobs to some location (station number) y_{il}. Note that the α_i, i=1,...,N will satisfy the usual traffic equations [6].

Theorem 1. The queueing network model described above has the following product form solution for its equilibrium state probabilities:

$$p(\mathbf{x}, \mathbf{y}) = C \prod_{i=1}^{N} f_i(\mathbf{x}_i)\, g_i(\mathbf{y}_i), \tag{2}$$

where $f_i(\mathbf{x_i})$ is given by

$$f_i(\mathbf{x}_i) = \begin{cases} \prod_{l=1}^{n_i} \frac{\alpha_i p_{i x_{il}}}{\mu_i}, & \text{if station } i \text{ is Type } I \\ n_i! \prod_{j=1}^{N} \left(\frac{\alpha_i}{\mu_i}\right)^{n_i} \frac{p_{ij}^{n_{ij}}}{n_{ij}!}, & \text{if station } i \text{ is Type } II \\ \prod_{j=1}^{N} \left(\frac{\alpha_i}{\mu_i}\right)^{n_i} \frac{p_{ij}^{n_{ij}}}{n_{ij}!}, & \text{if station } i \text{ is Type } III \\ \prod_{l=1}^{n_i} \frac{\alpha_i p_{i x_{il}}}{\mu_{i x_{il}}}, & \text{if station } i \text{ is Type } IV \end{cases} \tag{3}$$

and

$$g_i(\mathbf{y}_i) = \prod_{l=1}^{b_i} \frac{\alpha_{y_{il}} p_{y_{il} i}}{\mu_{y_{il}}} \tag{4}$$

where b_i denotes the total number of jobs blocked by station i and also C is the normalization constant:

$$C = [\sum_{\mathbf{x},\mathbf{y}\in\mathcal{S}} \prod_{i=1}^{N} f_i(\mathbf{x_i}) g_i(\mathbf{y_i})]^{-1}.$$

Proof. We restrict our proof to the case where jobs are only blocked at Type IV stations. Note that it is straightforward to see that the results can be extended to cases where several stations can cause blocking at the same time. The blocking event may happen as follows: When a job arrives at a station of Type IV, it will preempt the job in service; it will also check the destination station that it will visit next, to see whether it is full or has at least one buffer space available. If the next station is full then any job currently being served in any other station that will join the station causing blocking, will also be preempted. The preempted job will be put in a waiting list to reenter that station later.

For state $(\mathbf{x},\mathbf{y})$ the flow-out rate is $\sum_{i=1}^{N} \mu_i(x_i)$ for $n_i \geq 1$, and $\mu_i(x_i)$ depends on the type of station i. We now check the global balance equation for a given state $(\mathbf{x},\mathbf{y})$, The following states are flow-in states into state $(\mathbf{x},\mathbf{y})$.

State $(\mathbf{x} + x_{i0} - x_{x_{i0}1}, \mathbf{y})$: A job from station i moves to station x_{i0}, and chooses station $x_{x_{i0}1}$ as the station it will visit after station x_{i0}. Here $i \neq k$ and x_{i0} can be the station that satisfies $x_{x_{i0}1} \neq k$. The transition rate from this state to $(\mathbf{x},\mathbf{y})$ is $\mu_i p_{x_{i0},x_{x_{i0}1}}$. Thus

$$\begin{aligned} & p(\mathbf{x} + x_{i0} - x_{x_{i0}1}, \mathbf{y})\, \mu_i p_{x_{i0},x_{x_{i0}1}} \\ &= p(\mathbf{x},\mathbf{y}) \frac{\alpha_i p_{i,x_{i0}}}{\mu_i} \frac{\mu_{x_{i0}}}{\alpha_{x_{i0}} p_{x_{i0},x_{x_{i0}1}}} \mu_i p_{x_{i0},x_{x_{i0}1}} \\ &= p(\mathbf{x},\mathbf{y}) \frac{\mu_{x_{i0}}}{\alpha_{x_{i0}}} \alpha_i p_{i,x_{i0}}. \end{aligned} \tag{5}$$

State $(\mathbf{x} + x_{k0} - x_{x_{k0}1} - x_{y_{k0}n_{y_{k0}}}, \mathbf{y} + y_{x_{k0}})$, A job leaves station k which is causing blocking, moves to station x_{k0} and chooses station $x_{x_{k0}1}$ as its next station. We assume $x_{x_{k0}1} \neq k$. Since station k is the one that causes blocking, when a job moves out of this station, a buffer space becomes available, and one of the blocked jobs is released from its blocked situation and can be put back to the waiting queue, y_{k0}. We note that $x_{y_{k0}n_{y_{k0}}} = k$. The transition rate is $\mu_k p_{x_{k0},x_{x_{k0}1}}$. Thus

$$\begin{aligned} & p\left(\mathbf{x} + x_{k0} - x_{x_{k0}1} - x_{y_{k0}n_{y_{k0}}}, \mathbf{y} + y_{x_{k0}}\right) \mu_k p_{x_{k0},x_{x_{k0}1}} \\ &= p(\mathbf{x},\mathbf{y}) \frac{\alpha_k p_{k,x_{k0}}}{\mu_k} \frac{\mu_{x_{k0}}}{\alpha_{x_{k0}} p_{x_{k0},x_{x_{k0}1}}} \frac{\mu_{y_{k0}}}{\alpha_{y_{k0}} p_{y_{k0},x_{y_{k0}n_{k0}}}} \frac{\alpha_{y_{k0}} p_{y_{k0},k}}{\mu_{y_{k0}}} \mu_k p_{x_{k0},x_{x_{k0}1}} \\ &= p(\mathbf{x},\mathbf{y}) \frac{\mu_{x_{k0}}}{\alpha_{x_{k0}}} \alpha_k p_{k,x_{k0}} \end{aligned} \tag{6}$$

State $(\mathbf{x} + x_{i0} - x_{x_{i0}1} + x_{y_{kb_k}0}, \mathbf{y} - y_{kb_k})$. This transition is due to a departing job from station i, $i = 1, \ldots, N$, moving to station x_{i0} and choosing the next station $x_{x_{i0}1} = k$. The transition rate is $\mu_i p_{x_{i0}k}$. Due to the service discipline this job will get service immediately; because of the finite buffer capacity at station k, one of the jobs currently in service will be pre-empted and put in the waiting queue. Thus

$$\begin{aligned} & p\left(\mathbf{x} + x_{i0} - x_{x_{i0}1} + x_{y_{kb_k}0}, \mathbf{y} - y_{kb_k}\right) \mu_i p_{x_{i0}k} \\ &= p(\mathbf{x},\mathbf{y}) \frac{\alpha_i p_{ix_{i0}}}{\mu_i} \frac{\mu_{x_{i0}}}{\alpha_{x_{i0}} p_{x_{i0},x_{x_{i0}1}}} \frac{\alpha_{y_{kb_k}} p_{y_{kb_k},k}}{\mu_{y_{kb_k}}} \frac{\mu_{y_{kb_k}}}{\alpha_{y_{kb_k}} p_{y_{kb_k},k}} \mu_i p_{x_{i0}k} \\ &= p(\mathbf{x},\mathbf{y}) \frac{\mu_{x_{i0}}}{\alpha_{x_{i0}}} \alpha_i p_{ix_{i0}} \end{aligned} \tag{7}$$

Using equations (5), (6), (7) we sum over all feasible states and obtain:

$$\begin{aligned} & (5) + (6) + (7) = \\ & \quad p(\mathbf{x},\mathbf{y}) \sum_{i \neq k} \sum_{x_{i0} : x_{x_{i0}1} \neq k} \frac{\mu_{x_{i0}}}{\alpha_{x_{i0}}} \alpha_i p_{i,x_{i0}} \\ & + \; p(\mathbf{x},\mathbf{y}) \sum_{x_{i0} : x_{x_{i0}1} \neq k} \frac{\mu_{x_{k0}}}{\alpha_{x_{k0}}} \alpha_k p_{k,x_{k0}} \\ & + \; p(\mathbf{x},\mathbf{y}) \sum_{i=1}^{N} \sum_{x_{i0} : x_{x_{i0}1} = k} \frac{\mu_{x_{i0}}}{\alpha_{x_{i0}}} \alpha_i p_{ix_{i0}} \\ & = \; p(\mathbf{x},\mathbf{y}) \left(\sum_{x_{i0} : x_{x_{i0}1} \neq k} \mu_{x_{i0}} + \sum_{x_{i0} : x_{x_{i0}1} = k} \mu_{x_{i0}} \right) \end{aligned} \tag{8}$$

QED

3 Queueing Networks with Impatient Jobs

In certain communication networks, the fact that a job's service does not take place by a certain time, implies that it is routed to some other station. An example of this can be found in the TYMNET environment. In TYMNET an *overload condition* is said to occur if it takes a station more than 0.5sec to service a virtual circuit, in which case the job involved in the overload is immediately routed to another station.

Models inspired by this behaviour have been studied previously (see for instance [13]). Our model differs from the one examined in [13] in that we assume the limited time interval (LTI) begins when the job starts to receive service, rather than from the time the job enters a queue.

If the "impatient" jobs were routed according to the same routing probability as that of jobs which have terminated their service, then the effect of impatience would be merely to modify the effective service time in the queueing network. However, in this section we assume that impatient

jobs are routed according to a *distinct* routing probability, which is, in general, different from that of normally terminating jobs. We also assume that the LTI is an exponentially distributed random variable which depends on the server and on the job class.

If the network has no blocking, interestingly enough this does lead us to a special instance of G-Nets [12] with triggered job movement [12]. Indeed, in [12] one of us introduced a class of queueing networks where some jobs (called signals) have the ability of instantaneously moving a job from one station to another with a routing probability matrix which is distinct from the one used by jobs which have finished their service at a station. In the special case where these signals simply arrive to a station from the outside world according to a Poisson process, we are simply modelling the effect of the exponentially distributed LTI. The fact that we consider here finite buffer capacities, makes the results we present in this section distinct from those discussed in [12].

We consider here a queueing network with N stations, a total of K jobs and R job classes. Each station may be of the four Types as described in Section 1. The service time of class r jobs in station i is assumed to be exponentially distributed with parameter $1/\mu_{ir}$. After service normally ends, a job of class r at station i proceeds to station j and joins class s with probability $p_{ir;js}$.

The LTI of class r jobs in station i follows an exponential distribution of parameter ν_{ir} depending on the station i and the class r of the job. The routing probability of impatient jobs, i.e. those whose service has been interrupted by the LTI, is $q_{ir,js}$.

Stations have finite buffer capacity and blocking may occur only with the blocking-after-service discipline [1,2,3]. B_i denotes the finite buffer capacity of station i. We assume that the number of jobs in the system must satisfy the deadlock-free condition [5] given in (1), and that for each departure event there are no more than two jobs which move concurrently. Furthermore, *all blocked jobs must be located at Type II or at Type III stations.*

The traffic equations for the model are:

$$\alpha_{ir} = \sum_{j=1}^{N}\sum_{s=1}^{R}\gamma_{js}\mu_{js}p_{js,ir}, \tag{9}$$

$$\beta_{ir} = \sum_{j=1}^{N}\sum_{s=1}^{R}\gamma_{js}\nu_{js}q_{js,ir}, \tag{10}$$

where γ_{js} is a constant defined as

$$\gamma_{js} = \frac{\alpha_{js}+\beta_{js}}{\mu_{js}+\nu_{js}}.$$

Note that for Type I station we will have $\mu_{js} = \mu_j$ and $\nu_{js} = \nu_j$, for all $s = 1,\ldots,R$, in (9) and (10), i.e., the service rates are independent of the classes.

It can be shown that this network is not reversible [16] so that local balance equations do not hold. Our state space structure represents the nonblocked jobs and the blocked jobs separately, where the structure of the equilibrium state probability of the nonblocked jobs has exactly the same

form as that of a nonblocking system. The location of the blocked jobs and their class must be represented in the state-space.

We suppose that a job will choose its destination station and class when it finishes its service. We also assume that the blocked jobs will join that station which caused blocking according to a $First - Come - First - In$ $(FCFI)$ scheduling discipline.

The state of the system is denoted by $(\mathbf{x}, \mathbf{y})$, where $\mathbf{x} = (x_1, ..., x_N)$ and $\mathbf{y} = (\mathbf{y_1}, \ldots, \mathbf{y_N})$, $\mathbf{y_i} = (y_{i1}, \ldots, y_{ib_i})$. For Type I and IV stations we let $x_i = (x_{i1}, ..., x_{in_i})$, where $x_{il} \in \{1, R\}$ (for $l = 1, .., n_i$) denotes the class of the nth job, and n_i denotes the number of jobs in station i. For Type II and III stations, the state x_i is denoted $x_i = (n_{i1}, ..., n_{iR})$ where n_{ir} (for $r = 1, ..., R$) is the number of class r jobs in station i. Moreover, we have $y_{il} = \left(y_{il}^a, y_{il}^b\right)$ with $y_{il}^a \in \{1, \ldots, N\}$, and $y_{il}^b \in \{1, \ldots, R\}$. We use y_{il}^a and y_{il}^b to carry the information of the lth job blocked by station i, i.e., its location and class where y_{il}^b is the class that the blocked job will be when it enters the station that blocks it.

Thus, we have

$$K = \sum_{i=1}^{N} n_i + b, \tag{11}$$

with

$$0 \leq b \leq K - \min_{1 \leq i \leq N} \{B_i\}. \tag{12}$$

Theorem 2. The queueing network model described above has the following product form solution for its equilibrium state probabilities:

$$p(\mathbf{x}, \mathbf{y}) = C \prod_{i=1}^{N} f_i(\mathbf{x_i})\, g_i(\mathbf{x_i}, \mathbf{y_i}), \tag{13}$$

where $f_i(\mathbf{x_i})$ is defined as

$$f_i(\mathbf{x_i}) = \begin{cases} \prod_{l=1}^{n_i} \gamma_{ix_{il}} & \text{if station } i \text{ is Type I} \\ n_i! \prod_{r=1}^{R} \frac{1}{n_{ir}!} \gamma_{ir}^{n_{ir}}, & \text{if station } i \text{ is Type II} \\ \prod_{r=1}^{R} \frac{1}{n_{ir}!} \gamma_{ir}^{n_{ir}}, & \text{if station } i \text{ is Type III} \\ \prod_{l=1}^{n_i} \gamma_{ix_{il}} & \text{if station } i \text{ is Type IV} \end{cases} \tag{14}$$

and

$$g_i(\mathbf{x_i}, \mathbf{y_i}) = \prod_{l=1}^{b_i} \frac{\sum_{t=1}^{R} \left(\alpha_{y_{il}^a t} p_{y_{il}^a t, i y_{il}^b} + \beta_{y_{il}^a t} q_{y_{il}^a t, i y_{il}^b}\right)}{\mu_i(\mathbf{x_i}, \mathbf{y_i}) + \nu_i(\mathbf{x_i}, \mathbf{y_i})}, \tag{15}$$

The numerator of the right side of (15) denotes that given station i is full and causes blocking, the lth blocked job is staying at station y_{il}^a and when this job comes into station i, it will be a class

y^b_{il} job. Thus, we need to know the job's previous class. The denominator of the right side of (15) is defined as:

$$\mu_i(\mathbf{x_i}, \mathbf{y_i}) = \begin{cases} \mu_i & \text{if } i \text{ is Type } I \text{ station} \\ \sum_{t=1}^{R} \frac{n_{it}}{n_i} \mu_{it} & \text{if } i \text{ is Type } II \text{ station} \\ \sum_{t=1}^{R} n_{it} \mu_{it} & \text{if } i \text{ is Type } III \text{ station} \\ \mu_{iy^b_{i1}} & \text{if } i \text{ is Type } IV \text{ station.} \end{cases} \tag{16}$$

Note that in (16) there is a difference between Type II, III and IV stations where the service rate depends on the job class. For Type II and III stations, $\mu_i(\mathbf{x_i}, \mathbf{y_i})$ depends on $\mathbf{x_i}$, but for Type IV it depends on $\mathbf{y_i}$. Note that $\nu_i(\mathbf{x_i}, \mathbf{y_i})$ in (15) is identical to (16), except that the μ_i's are replaced by ν_i's.

We only sketch the proof which is similar to the one given in [3]. It follows from the following argument: we may consider the buffers occupied by the blocked jobs which are stored in Type II or III stations only, as the extended buffers of the stations causing blocking. The existence of blocked jobs in Type II or Type III stations will not affect the service of the non-blocking jobs in these stations. That is the throughput rate of these two types of stations will be dependent on the number of non-blocked jobs in this station only. Thus, we can ignore the locations of the blocked jobs and the result will follow.

4 Gelenbe Networks and Repetitive-Service-Blocking

Recently, one of us [7-11] discovered a new class of queueing network models involving "negative" jobs. In these queueing networks there are positive and negative jobs. A positive job acts as the normal job in classical queueing networks. A negative job however, when it enters a station will instantly destroy one positive job and will itself be simultaneously destroyed. In other words, a negative and positive job in a station instantly cancel each other and disappear from the network. Note that the model is only of interest for open networks since negative and positive jobs have a tendency to cancel each other.

The underlying traffic quations of this model are non-linear, and it was shown that [7-11] these models have exact product form solution. In [12] a generalization of this idea to "triggerred job movement" was introduced, whereby the network contains normal jobs and "signals". A signal has the power of instantaneously moving a normal job from one station to another, or it can destroy it — in which case the signal is acting as a negative job. Again in [12], one of us showed that these networks have product form. These models can be applied in a resource request environment where negative jobs are commands to cancel previous requests (positive jobs), or signals which are used

to carry out routing decisions.

In the present section, we will consider networks with N stations of Type II and of Type IV, containing R classes of positive jobs and one class of negative jobs. The positive jobs of class r arrive at the ith station from outside according to a Poisson process with rates λ_{ir} and negative jobs arrive at the ith station according to a Poisson process with rate ν_i. A positive job of class r leaving station i will enter station j and join class s as a positive job with probability $p_{ir;js}$. Since the system is open, we also assume the probability that a job of class r will leave the system after its service in station i:

$$p_{ir,o} = 1 - \sum_{j=1}^{N} \sum_{s=1}^{R} p_{ir,js}.$$

Note that Gelenbe [7-11] assumes that a positive job may become a negative job via an appropriate routing matrix. However, in the model considered in this section we do not allow this. A positive job arriving at a station increases the queue length by one. A negative job arriving at a station will decrease the queue length by one if there is at least one positive job present in the station. A negative job arriving at an empty queue is cleared from the network. Whenever a negative job arrives at the system, it will kill one of the positive jobs in the station where it arrived. If the station is of Type IV, then the job in service will be killed. If the station is of Type II, each job has equal probability to be killed. Moreover, in our model the negative jobs arrive at station i and take the necessary actions there. They do not circulate in the network.

We are interested in the case of a network whose stations have finite capacities denoted by B_i. We investigate the repetitive-service-blocking type [4], that is, the blocking event occurs when upon completion of its service of a particular station's server, a job attempts to proceed to its next station. If at that moment, its destination station is full, the job is rejected. The job goes back to the server of the source station and immediately receives a new service. This is repeated until the next station releases a job and a place becomes available. Note that the job selects a destination independently after each round of service according to the routing probabilities. Deadlocks are impossible if the network is irreducible, (i.e., each station is reachable from every other station). As long as there is at least one free space in some station a job will eventually move into it even if this takes a long sequence of trials.

Let

$$\alpha_{ir} = \sum_{j=1}^{N} \sum_{s=1}^{R} \frac{\lambda_{js} + \alpha_{js}}{\mu_{js} + \nu_j} \mu_{js} p_{js,ir}. \tag{17}$$

We assume that the system is reversible:

$$\frac{\lambda_{ir} + \alpha_{ir}}{\mu_{ir} + \nu_i} \mu_{ir} p_{ir,js} = \frac{\lambda_{js} + \alpha_{js}}{\mu_{js} + \nu_j} \mu_{js} p_{js,ir} \tag{18}$$

and (17) becomes

$$\alpha_{ir} = \frac{\mu_{ir} \sum_{j=1}^{N} \sum_{s=1}^{R} p_{ir,js}}{\mu_{ir} p_{ir,o} + \nu_i}. \tag{19}$$

Theorem 3. The queueing network model described above has the following product form solution for equilibrium state probabilities:

$$p(\mathbf{x}) = C \prod_{i=1}^{N} f_i(\mathbf{x}_i), \tag{20}$$

where $f_i(\mathbf{x}_i)$ is defined as

$$f_i(\mathbf{x}_i) = \begin{cases} n_i! \prod_{r=1}^{R} \frac{1}{n_{ir}!} \left(\frac{\lambda_{ir} + \alpha_{ir}}{\mu_{ir} + \nu_i} \right)^{n_{ir}}, & \text{if station } i \text{ is Type II} \\ \prod_{l=1}^{n_i} \frac{\lambda_{ix_{il}} + \alpha_{ix_{il}}}{\mu_{ix_{il}} + \nu_i} & \text{if station } i \text{ is Type IV} \end{cases} \tag{21}$$

and C is the normalization constant.

Proof. Since the system is reversible, we check the detailed balance equations of the transition cases for each Type II and IV station:

Case 1: (Type II Stations)

Case 1.1: A job leaves station i and goes to station j, for $n_i < B_i$ and $n_j < B_j$.

$$\begin{aligned} & p(\mathbf{n} + e_{ir} - e_{js}) \left(\frac{n_{ir} + 1}{n_i + 1} \mu_{ir} p_{ir,js} \right) \\ &= p(\mathbf{n}) \frac{n_i + 1}{n_{ir} + 1} \frac{\lambda_{ir} + \alpha_{ir}}{\mu_{ir} + \nu_i} \frac{n_{js}}{n_j} \frac{\mu_{js} + \nu_j}{\lambda_{js} + \alpha_{js}} \left(\frac{n_{ir} + 1}{n_i + 1} \mu_{ir} p_{ir,js} \right) \\ &= p(\mathbf{n}) \frac{n_{js}}{n_j} \frac{\mu_{js} + \nu_j}{\lambda_{js} + \alpha_{js}} \frac{\lambda_{ir} + \alpha_{ir}}{\mu_{ir} + \nu_i} \mu_{ir} p_{ir,js} \\ &= p(\mathbf{n}) \frac{n_{js}}{n_j} \frac{\mu_{js} + \nu_j}{\lambda_{js} + \alpha_{js}} \frac{\lambda_{js} + \alpha_{js}}{\mu_{js} + \nu_j} \mu_{js} p_{js,ir} \\ &= p(\mathbf{n}) \frac{n_{js}}{n_j} \mu_{js} p_{js,ir}. \end{aligned} \tag{22}$$

Case 1.2: A job either leaves the system or is killed by a negative job.

$$\begin{aligned} & p(\mathbf{n} + e_{ir}) \left(\frac{n_{ir} + 1}{n_i + 1} \nu_i + \frac{n_{ir} + 1}{n_i + 1} \mu_{ir} p_{ir,o} \right) \\ &= p(\mathbf{n}) \frac{n_i + 1}{n_{ir} + 1} \frac{\lambda_{ir} + \alpha_{ir}}{\mu_{ir} + \nu_i} \left(\frac{n_{ir} + 1}{n_i + 1} \nu_i + \frac{n_{ir} + 1}{n_i + 1} \mu_{ir} p_{ir,o} \right) \\ &= p(\mathbf{n}) \frac{\lambda_{ir} + \alpha_{ir}}{\mu_{ir} + \nu_i} \left(\nu_i + \mu_{ir} - \sum_{j=1}^{N} \sum_{s=1}^{R} \mu_{ir} p_{ir,js} \right) \\ &= p(\mathbf{n}) \left[(\lambda_{ir} + \alpha_{ir}) - \sum_{j=1}^{N} \sum_{s=1}^{R} \frac{\lambda_{ir} + \alpha_{ir}}{\mu_{ir} + \nu_i} \mu_{ir} p_{ir,js} \right] \end{aligned}$$

$$
\begin{aligned}
&= p(\mathbf{n})\left[(\lambda_{ir}+\alpha_{ir}) - \sum_{j=1}^{N}\sum_{s=1}^{R}\frac{\lambda_{js}+\alpha_{js}}{\mu_{js}+\nu_j}\mu_{js}p_{js,ir}\right] \\
&= p(\mathbf{n})(\lambda_{ir}) \qquad (23)
\end{aligned}
$$

Case 1.3: A Job Arrives from Outside.

$$
\begin{aligned}
&p(\mathbf{n}-e_{ir})\,\lambda_{ir} \\
&= p(\mathbf{n})\frac{n_{ir}}{n_i}\frac{\mu_{ir}+\nu_i}{\lambda_{ir}+\alpha_{ir}}\lambda_{ir} \\
&= p(\mathbf{n})\frac{n_{ir}}{n_i}\left[(\mu_{ir}+\nu_i) - \frac{\mu_{ir}+\nu_i}{\lambda_{ir}+\alpha_{ir}}\alpha_{ir}\right] \\
&= p(\mathbf{n})\frac{n_{ir}}{n_i}\left[(\mu_{ir}+\nu_i) - \frac{\mu_{ir}+\nu_i}{\lambda_{ir}+\alpha_{ir}}\left(\sum_{j=1}^{N}\sum_{s=1}^{R}\frac{\lambda_{js}+\alpha_{js}}{\mu_{js}+\nu_j}\mu_{js}p_{js,ir}\right)\right] \\
&= p(\mathbf{n})\frac{n_{ir}}{n_i}\left[(\mu_{ir}+\nu_i) - \frac{\mu_{ir}+\nu_i}{\lambda_{ir}+\alpha_{ir}}\left(\sum_{j=1}^{N}\sum_{s=1}^{R}\frac{\lambda_{ir}+\alpha_{ir}}{\mu_{ir}+\nu_i}\mu_{ir}p_{ir,js}\right)\right] \\
&= p(\mathbf{n})\frac{n_{ir}}{n_i}\left[(\mu_{ir}+\nu_i) - \sum_{j=1}^{N}\sum_{s=1}^{R}\mu_{ir}p_{ir,js}\right] \\
&= p(\mathbf{n})\frac{n_{ir}}{n_i}(\mu_{ir}p_{ir,o}+\nu_i) \qquad (24)
\end{aligned}
$$

Case 2: (Type IV Stations) Here we use the lower index to indicate the earlier job that will get service or enter a blocking station.

Case 2.1: A job leaves station i and goes to station j, for $n_i < B_i$ and $n_j < B_j$. Since we treat Type IV stations we denote the class of that job in station j to be x_{j1}.

$$
\begin{aligned}
&p(\mathbf{x}+x_{i0}-x_{j1})\left(\mu_{ix_{i0}}p_{ix_{i0},jx_{j1}}\right) \\
&= p(\mathbf{x})\frac{\lambda_{ix_{i0}}+\alpha_{ix_{i0}}}{\mu_{ix_{i0}}+\nu_i}\frac{\mu_{jx_{j1}}+\nu_j}{\lambda_{jx_{j1}}+\alpha_{jx_{j1}}}\mu_{ix_{i0}}p_{ix_{i0},jx_{j1}} \\
&= p(\mathbf{x})\mu_{jx_{j1}}p_{jx_{j1},ix_{i0}} \qquad (25)
\end{aligned}
$$

Case 2.2: One job disappears from the system either by leaving the system or by being killed by a negative job,

$$
\begin{aligned}
&p(\mathbf{x}+x_{i0})(\mu_{ix_{i0}}p_{ix_{i0},o}+\nu_i) \\
&= p(\mathbf{x})\frac{\lambda_{ix_{i0}}+\alpha_{ix_{i0}}}{\mu_{ix_{i0}}+\nu_i}(\mu_{ix_{i0}}p_{ix_{i0},o}+\nu_i) \\
&= p(\mathbf{x})\left(\lambda_{ix_{i0}}+\alpha_{ix_{i0}} - \sum_{j=1}^{N}\sum_{s=1}^{R}\frac{\lambda_{ix_{i0}}+\alpha_{ix_{i0}}}{\mu_{ix_{i0}}+\nu_i}\mu_{ix_{i0}}p_{ix_{i0},js}\right) \\
&= p(\mathbf{x})\left(\lambda_{ix_{i0}}+\alpha_{ix_{i0}} - \sum_{j=1}^{N}\sum_{s=1}^{R}\frac{\lambda_{js}+\alpha_{js}}{\mu_{js}+\nu_j}\mu_{js}p_{js,ix_{i0}}\right) \\
&= p(\mathbf{x})(\lambda_{ix_{i0}}) \qquad (26)
\end{aligned}
$$

Case 2.3: Arrival Occurs.

$$
\begin{aligned}
& p(\mathbf{x}-x_{i1})\,\lambda_{ix_{i1}} \\
&= p(\mathbf{x})\,\frac{\mu_{ix_{i1}}+\nu_i}{\lambda_{ix_{i1}}+\alpha_{ix_{i1}}}\lambda_{ix_{i1}} \\
&= p(\mathbf{x})\left[(\mu_{ix_{i1}}+\nu_i)-\frac{\mu_{ix_{i1}}+\nu_i}{\lambda_{ix_{i1}}+\alpha_{ix_{i1}}}\alpha_{ix_{i1}}\right] \\
&= p(\mathbf{x})\left[(\mu_{ix_{i1}}+\nu_i)-\frac{\mu_{ix_{i1}}+\nu_i}{\lambda_{ix_{i1}}+\alpha_{ix_{i1}}}\sum_{j=1}^{N}\sum_{s=1}^{R}\frac{\lambda_{js}+\alpha_{js}}{\mu_{js}+\nu_j}\mu_{js}p_{js,ix_{i1}}\right] \\
&= p(\mathbf{x})\left[(\mu_{ix_{i1}}+\nu_i)-\frac{\mu_{ix_{i1}}+\nu_i}{\lambda_{ix_{i1}}+\alpha_{ix_{i1}}}\sum_{j=1}^{N}\sum_{s=1}^{R}\frac{\lambda_{ix_{i1}}+\alpha_{ix_{i1}}}{\mu_{ix_{i1}}+\nu_i}\mu_{ix_{i1}}p_{ix_{i1},js}\right] \\
&= p(\mathbf{x})\left[(\mu_{ix_{i1}}+\nu_i)-\sum_{j=1}^{N}\sum_{s=1}^{R}\mu_{ix_{i1}}p_{ix_{i1},js}\right] \\
&= p(\mathbf{x})\left(\mu_{ix_{i1}}p_{ix_{i1},o}+\nu_i\right) \qquad (27)
\end{aligned}
$$

QED

5 Conclusions

In this paper we have exhibited three models of queueing networks with blocking whose stationary probability distributions have product form. These models are inspired by the needs of computer performance analysis for computationally efficient solutions to models arising from computer-communication systems. Two of the models studied here are closely related to the novel queueing networks with new types of jobs ("negative jobs" and "signals") which have been recently introduced in [7-12] and which also have product form.

In the *first model* jobs have "early routing decisions": a job chooses its next station according to a Markovian rule as soon as it enters a station, and when it reaches the head of the queue it will block the server stopping it from serving other jobs if either one (or both) of its two downstream stations is full. In the *second model*, each job has a time limit (starting at the time it begins service) by which it must complete service. If the job is still in service when the time limit runs out, it will leave its current station and is then routed to another station according to a routing probability which is distinct from the one after normal service completion. Here blocking occurs after service when a job completing service cannot proceed to the next station if its buffer is full, and thus blocks its current server. The *third model* is derived from networks with "positive and negative jobs", and contains a single negative and multiple positive job classes. Each station has finite buffer capacity and jobs can be blocked with "repetitive-service-blocking" if the station they need to go to is full.

Our conclusion is, in fact, a question which has intrigued queueing theorists and computer

performance analysts for a long time, and which does not seem to be close to receiving an adequate answer: *What is it that provides a queueing network with the product form property?*

It appears that the more we discover new product forms, the further we move away from a simple answer to this question.

REFERENCES

[1] I. F. Akyildiz, "Exact Product Form Solution for Queueing Networks with Blocking", *IEEE Transactions on Computers*, Vol. **C-36**, No. 1, January 1987, pp. 122-125.

[2] I. F. Akyildiz and H. von Brand, "Exact Solutions for Networks of Queues with Blocking-After-Service", *Technical Report, Georgia Tech, College of Computing, April 1991, GIT-COC-91-26.*

[3] I.F. Akyildiz and C.C. Huang, "Exact Analysis of Queueing Networks with Multiple job Classes and Blocking-After-Service", to appear in *Queueing Systems: Theory and Applications: QUESTA*, 1993.

[4] I. F. Akyildiz and H. von Brand, "Computational Algorithms for Networks of Queues with Rejection Blocking", *Acta Informatica*, Vol. 26, pp. 559-576, July 1989.

[5] I. F. Akyildiz and J. Liebeherr, "Optimal Deadlock Free Buffer Allocation in Multiple Chain Blocking Networks of Queues", *Proc. of Performance of Distributed Systems*, North-Holland, September 1991, pp. 245-264.

[6] F. Baskett, K. M. Chandy, R. R. Muntz and F. G. Palacios, "Open, Closed and Mixed Networks of Queues with Different Classes of Customers", *Journal of the ACM*, Vol. **22**, Vol. 15, pp. 248-260, April 1975.

[7] J.M. Fourneau and E. Gelenbe, "Networks with Negative Customers and Classes of Positive Customers", EHEI Technical Report 91-6, University of Paris V, July 1991.

[8] E. Gelenbe, "Queueing Networks with Negative and Positive Customers and Product Form Solutions", *Journal of Applied Probability*, Vol. **28**, pp. 656-663, 1991.

[9] E. Gelenbe, P. Glynn and K. Sigman, "Queues with Negative Arrivals", *Journal of Applied Probability*, Vol. **28**, pp. 245-250, 1991.

[10] E. Gelenbe and R. Schassberger, "Stability of G-Networks", *Probability and Its Applications in the Engineering and Information Sciences*, Vol. **6**, pp. 271-276, 1993.

[11] E. Gelenbe, "G-Networks: Negative Customers with Batch Removal", submitted to *Performance Evaluation Journal*, 1992.

[12] E. Gelenbe, "G-Networks with Triggered Customer Movement", accepted for publication in *Journal of Applied Probability*, 1993.

[13] E.A. Kovalev, "Behavior of Impatient Requests in Queueing Network with Limited Queueing Time", *Avtomatika i Vychislitel'naya Tekhnika*, Vol. **21**, No. 2, pp. 88-90, 1987.

[14] F. P. Kelly, "Networks of Queues with Customers of Different Types", *Journal of Applied Probability*, Vol. **12**, pp. 542-554, 1975.

[15] F. P. Kelly, "Networks of Queues", *Advances in Applied Probability*, Vol. **8**, No. 2, pp. 416-432, June 1976.

[16] F. P. Kelly, *Reversibility and Stochastic Networks*, Wiley, London, 1979.

Exact Analysis of Closed Queueing Networks with Kanban Blocking

Ian F. Akyildiz

School of Electrical Engineering

Georgia Institute of Technology

Atlanta, GA 30332

March 2, 1993

Abstract

An exact product form solution is obtained for a queueing network with Type I, II, III and IV stations having finite buffer capacities. A job is blocked after service if its destination is full but the server continues to serve the unblocked jobs in its station. The service time distributions are assumed to be exponential for Type I and Coxian for Type II, III, and IV. It is also assumed that in the beginning of the service each phase of the Coxian representation may be reached with a positive probability. The steady-state distribution of the number of blocked and unblocked jobs depends on the rates of the Cox phases. Thus, the insensitivity property does not hold in this model in contrast to the classical BCMP queueing networks. A convolution algorithm is developed for the computation of the normalization constant. New formulae are derived for the computation of performance measures.

Key Words: Queueing Networks, Product Form Solution, Blocking Mechanisms, Minimal Blocking, Cox Distributions, Insensitivity Property, Convolution Algorithm [1]

---

[1]This work was supported by National Science Foundation (NSF) under Grant No. CCR-90-11981.

1 Introduction

Performance has been a major issue in the design and implementation of systems such as computer systems, production systems, flexible manufacturing systems and communication networks. Thus, tools and techniques for predicting performance measures are of great interest. The area of exact computational algorithms for product-form queueing networks may be viewed as being one of the cornerstones of queueing network analysis [6,7,8]. The stations in the BCMP networks [6,7,8] may be one of the following types: a single server using a first come first served queueing discipline (FCFS) or so-called Type I, a single server with a processor sharing queueing discipline (PS) or Type II, an infinite number of servers (IS) or Type III, and a single server scheduled according to the preemptive resume last come first served discipline (LCFS-PR) or Type IV. However, this class of product form queueing networks [6,7,8] is restricted to stations with infinite buffer capacities and cannot provide much insight into the phenomenon of blocking which is our main objective in this paper. The set of rules that dictates when a station becomes blocked and when it becomes unblocked is commonly referred to as the *blocking mechanism.* There are basically only a few blocking mechanisms that have been extensively studied in the literature [2,12,14]. In this paper we investigate the so-called *kanban blocking* mechanism, in short form KB, also named as *minimal blocking* [9,10], i.e., a job upon service completion at station i attempts to join destination station j. If station j is full, the job is forced to wait in the station i until a buffer becomes available in the destination station. The server (or servers) is not blocked and continues to serve the unblocked jobs. This blocking mechanism is shown to be equivalent to the *kanban policy* [11,16] which is used in Japanese manufacturing lines. In fact for Type II and Type III BCMP stations, the kanban blocking is identical to the *blocking-after-service* (BAS) mechanism since the servers in these cases continue to serve the unblocked jobs.

Few exact computational algorithms are known about queueing networks with blocking [2,12,14]. The most studied case is the cyclic network with two stations with finite buffers [2,12]. In particular, Akyildiz [1] demonstrated that such network with K jobs are equivalent to a nonblocking network, with the same parameters as the blocking network but with infinite

capacities and with K' jobs, where $K' = min(K, B_1+1)+min(K, B_2+1)-K$ and B_i is the buffer capacity of station i. Onvural and Perros [13] showed that a network with a limited number of jobs $K = min_{i\in N}(B_i) + 1$ and BAS blocking mechanism has a product-form solution. This result has been extended by Akyildiz and von Brand [3] to a multiple job class queueing network with blocking under the same condition on the number of jobs in the network.

Akyildiz and Huang [4] obtained a product form solution for a queueing network with Type II and Type III BCMP stations and finite capacities, multiple job classes, BAS blocking mechanism and exponential service time distributions. They assume that no more than one blocking may occur in the network at a time. So, the assumption on the number of jobs in the network given in [3,13] is relaxed.

In this work we analyze Type I, II, III, and IV BCMP stations and kanban blocking mechanism. This work can be also related to [9,10] where Mitra and Mitrani analyze a tandem queueing network with Type I stations and kanban blocking mechanism using an iterative procedure. To obtain a product form solution, we assume that two full stations do not interact with each other directly, but we allow more than one blocking at a time. The service time distributions are assumed to be exponential for Type I stations and Coxian for Type II, III, and IV stations. In contrast to the BCMP product form solution, a necessary condition on the Cox distribution representations is needed. The condition is that each phase of the Coxian representation may be reached in the beginning of the service with a positive probability. The number of blocked and unblocked jobs steady-state distribution depends on the rates of the Cox phases. Thus, the insensitivity property does not hold for queueing networks with KB mechanism in contrast to classical BCMP queueing networks [6]. In fact, this property was already recognized in [3] where we showed that the insensitivity property does not hold for two-station queueing networks with blocking-after-service mechanism.

The paper is organized as follows: in section 2 we describe the network model in detail. In section 3 we state the product form solution, and prove it. In section 4 we determine the steady-state distribution of the number of blocked and unblocked jobs. In section 5 we develop a convolution algorithm for the computation of the normalization constant. In

section 6 we derive new formulae for the computation of performance measures in this type of models. Finally in section 7 we conclude the paper.

2 Model Description

We consider a closed queueing network with N stations and K single class jobs. The routing matrix, denoted by $P = (p_{ij})_{1 \leq i,j \leq N}$ is assumed to be irreducible. Each station has a finite buffer capacity denoted by $(B_i)_{1 \leq i,j \leq N}$. The scheduling discipline in each station is either FCFS, PS, IS, or LCFS-PR, with finite buffers. Upon a completion of a service at station i, a job attempting to enter a full station j ($p_{ij} > 0$) is forced to wait in station i until it is allowed to enter station j. The server is not blocked and continues to serve the unblocked jobs. Jobs blocked by the same station j are unblocked in first blocked first unblocked order. This blocking discipline is known as *kanban blocking* (KB) in the literature [9,10].

The service time distributions are assumed to be Coxian. We denote by S_i the number of phases of the service time distribution at station i and by μ_{im} its rate at the m^{th} Coxian-phase ($1 \leq m \leq S_i$). This service time distribution can be represented by a network of exponential stages where f_{im} is the probability that a job starts its service time requirement at the m^{th} phase ($\sum_{m=1}^{S_i} f_{im} = 1$), d_{im} is the probability that a job completing the phase m leaves the i^{th} station, and a_{im} is the probability that a job completing the phase m proceeds to phase $m+1$ where ($a_{im} = 1 - d_{im}$). However, for Type I stations the service time distributions are assumed to be exponential. Hereafter, we will use the same notations for Type I stations and for Type II, III, and IV stations but with $S_i = 1$. We denote $(A_{im})_{1 \leq m \leq S_i}$ as the probability that a job at station i may reach the m^{th} phase of its service which is obtained by recurrence:

$$A_{i1} = f_{i1} \tag{1}$$

$$A_{im+1} = A_{im} a_{im} + f_{im+1} \quad for \quad 1 \leq m \leq S_i - 1. \tag{2}$$

We denote by k_i ($1 \leq i \leq N$) the number of unblocked jobs in station i, by k_{im} ($1 \leq m \leq S_i$) the number of jobs in station i at the m^{th} phase. We denote also by b_i the number of jobs blocked by station i, and by b_{ij} the number of jobs blocked in station j by station

i ($b_i = \sum_{j=1}^{N} b_{ij}$). The vector $\bar{n}_i = (n_{i1}, \ldots, n_{ib_i})$ is the ordered set of station index of jobs blocked by station i. For a station i of Type IV, we denote by t_{ik} $(1 \leq k \leq k_i)$ the phase of the k^{th} job in station i according to LCFS order. The state $\hat{k}_i$ of station i is defined by

$$\hat{k}_i = (\bar{k}_i, \bar{n}_i)$$

where

$$\bar{k}_i = \begin{cases} k_i & \text{if } i \text{ is a Type I station} \\ (k_{i1}, \ldots, k_{iS_i}) & \text{if } i \text{ is a Type II or III station} \\ (t_{i1}, \ldots, t_{ik_i}) & \text{if } i \text{ is a Type IV station} \end{cases}$$

The state of the network is given by:

$$\hat{k} = (\hat{k}_1, \ldots, \hat{k}_N).$$

According to the queueing discipline in station i, the service rates may depend on k_i, k_{im}, and on the phase of the first job (in Type IV stations). These rates will be denoted by $\mu_{im}(\bar{k}_i)$:

$$\mu_{im}(\bar{k}_i) = \begin{cases} \mu_i & \text{if } i \text{ is a Type I station and } k_i > 0 \\ 0 & \text{if } i \text{ is a Type I station and } k_i = 0 \\ \mu_{im}(k_i, k_{im}) = \frac{k_{im}}{k_i}\mu_{im} & \text{if } i \text{ is a Type II station} \\ \mu_{im}(k_i, k_{im}) = k_{im}\mu_{im} & \text{if } i \text{ is a Type III station} \\ \mu_{im} & \text{if } i \text{ is a Type 4 station and } t_{i1} = m \\ 0 & \text{if } i \text{ is a Type IV station and } t_{i1} \neq m \end{cases}$$

We assume that the network is deadlock-free [5] and two full stations do not interact directly, i.e.,

$$K < min_{[1 \leq i,j \leq N,\ p_{ij} > 0]}(B_i + B_j). \tag{3}$$

In addition, we also assume that the parameters of the Coxian distributions satisfy

$$f_{im} > 0, \quad 1 \leq i \leq N, \quad 1 \leq m \leq S_i. \tag{4}$$

3 Product Form Solution

We describe the transitions which may occur in the network and may lead to a predetermined state $\hat{k}$. For the case where all stations are not full, i.e., $k_i < B_i, \quad 1 \leq i \leq N$, the transitions are the same as in the underlying BCMP network. Let us now assume that there are β full stations. The set of these stations is denoted by Γ. So,

$$k_l = \sum_{m=1}^{S_l} k_{lm} = B_l, \quad for \quad l \in \Gamma.$$

Four different types of transitions may lead to the state $\hat{k}$. In order to simplify the description of the transitions, we convene that for Type I stations $a_{im} = 0$ for $m \neq 0$ because there is only one phase in this case. When a job moves from a station of Type IV, it is the job who has been in service (first position) and when a job moves to a station of Type IV, it is the job who is going to be in service.

[a] *Phase Transitions:* A job in station i moves from phase m to phase $m+1$. The initial state has to be $T^i_{m+1,m}(\hat{k})$ where $T^i_{m+1,m}(\hat{k})$ is the same state as $\hat{k}$ with one more job at the m^{th} phase in station i and one less job at the $(m+1)^{th}$ phase in station i. So, when the transition occurs we obtain the state $\hat{k}$. This transition occurs with rate $\mu_{im}(T^i_{m+1,m}(\hat{k}))a_{im}$.

[b] *Transitions Independent of Full Stations:* A job moves from phase m at the i^{th} station ($i \notin \Gamma$) to the n^{th} phase at the j^{th} station ($j \notin \Gamma$). The initial state has to be $T^{ji}_{n,m}(\hat{k})$ where $T^{ji}_{n,m}(\hat{k})$ is the same state as $\hat{k}$ with one more job at the m^{th} phase in station i and one less job at the n^{th} phase in station j. This transition occurs with rate $\mu_{im}(T^{ji}_{n,m}(\hat{k}))d_{im}f_{jn}p_{ij}$.

[c] *Transitions out of Full Stations:* A job moves from phase m at the i^{th} station ($i \in \Gamma$) to the n^{th} phase at the j^{th} station ($j \notin \Gamma$). The first job blocked by station i moves to the p^{th} phase in station i. The blocked job may belong to any station l ($1 \leq l \leq N$). The initial state has to be $T^{jil}_{n,m,p}(\hat{k})$ where $T^{jil}_{n,m,p}(\hat{k})$ is the same state as $\hat{k}$ with one more job in station l blocked by station i in the first position, one less job at the p^{th}

phase in station i, one more job at the m^{th} phase in station i and one less job in the n^{th} phase at the j^{th} station. The transition rate is $\mu_{im}(T^{jil}_{n,m,p}(\hat{k}))d_{im}f_{jn}f_{ip}p_{ij}$.

[d] *Transitions into Full Stations:* A job moves from phase m at the i^{th} station ($i \notin \Gamma$) to the last blocked position by station ($j \in \Gamma$). The initial state has to be $T^{\cdot i,j\cdot}_{m}(\hat{k})$ where $T^{\cdot i,j\cdot}_{m}(\hat{k})$ is as $\hat{k}$ with one more job in the m^{th} phase in station i and minus the last blocked job. i is the station index of the last job blocked by station j, i.e., $i = n_{jb_j}$. The transition rate is $\mu_{im}(T^{\cdot i,j\cdot}_{m}(\hat{k}))d_{im}p_{ij}$.

We denote by $(e_i)_{1\leq i\leq N}$ a solution of the traffic equation:

$$e_i \;=\; \sum_{j=1}^{N} e_j p_{ji} \quad 1 \leq i \leq N. \tag{5}$$

Theorem 1 *The steady-state distribution of the network is given by the following product form solution:*

$$P(\hat{k}) = \gamma \prod_{i=1}^{N} F_i(\hat{k}_i), \tag{6}$$

where γ is the normalization constant and $F_i(\hat{K}_i)$ is a function defined by

$$F_i(\hat{k}) = \begin{cases} (e_i\mu_i^{-1})^{k_i} \prod_{j=1}^{N} \left(\frac{e_j p_{ji}}{\sum_{m=1}^{S_i} \frac{f_{im}}{A_{im}} \mu_{im}(\bar{k}_i)}\right)^{b_{ij}} & \text{if } i \text{ is a Type I station} \\ k_i! \prod_{m=1}^{S_i} \frac{(e_i A_{im} \mu_{im}^{-1})^{k_{im}}}{k_{im}!} \prod_{j=1}^{N} \left(\frac{e_j p_{ji}}{\sum_{m=1}^{S_i} \frac{f_{im}}{A_{im}} \mu_{im}(\bar{k}_i)}\right)^{b_{ij}} & \text{if } i \text{ is a Type II station} \\ \prod_{m=1}^{S_i} \frac{(e_i A_{im} \mu_{im}^{-1})^{k_{im}}}{k_{im}!} \prod_{j=1}^{N} \left(\frac{e_j p_{ji}}{\sum_{m=1}^{S_i} \frac{f_{im}}{A_{im}} \mu_{im}(\bar{k}_i)}\right)^{b_{ij}} & \text{if } i \text{ is a Type III station} \\ \prod_{k=1}^{k_i} (e_i A_{it_{ik}} \mu_{it_{ik}}^{-1}) \prod_{j=1}^{N} \left(\frac{e_j p_{ji}}{\sum_{m=1}^{S_i} \frac{f_{im}}{A_{im}} \mu_{im}(\bar{k}_i)}\right)^{b_{ij}} & \text{if } i \text{ is a Type IV station} \end{cases}$$

Proof.

For a state where all stations are not full, the steady-state distribution and the global balance equation are the same as in the underlying BCMP network [6]. So, the proof in this case is identical to the proof in the BCMP network. We consider now the case where $\beta \geq 1$ stations are full. The global balance equation, according to the transitions described above is given by

$$P(\hat{k}) \sum_{i=1}^{N} \sum_{m=1}^{S_i} \mu_{im}(\bar{k}_i) \;=\; \sum_{i=1}^{N} \sum_{m=1}^{S_i} P(T^{i}_{m+1,m}(\hat{k}))\mu_{im}(T^{i}_{m+1,m}(\hat{k}))a_{im}$$

$$+\sum_{i\notin\Gamma}\sum_{m=1}^{S_i}\sum_{j\notin\Gamma}\sum_{n=1}^{S_j} P(T^{ji}_{n,m}(\hat{k}))\mu_{im}(T^{ji}_{n,m}(\hat{k}))b_{im}f_{jn}p_{ij}$$

$$+\sum_{l=1}^{N}\sum_{i\in\Gamma}\sum_{m=1}^{S_i}\sum_{p=1}^{S_i}\sum_{j\notin\Gamma}\sum_{n=1}^{S_j} P(T^{jil}_{n,m,p}(\hat{k}))\mu_{im}(T^{jil}_{n,m,p}(\hat{k}))b_{im}f_{jn}f_{ip}p_{ij}$$

$$+\sum_{j\in\Gamma,i=n_{jb_j}}\sum_{m=1}^{S_i} P(T^{.i,j.}_{m}(\hat{k}))\mu_{im}(T^{.i,j.}_{m}(\hat{k}))b_{im}p_{ij}. \qquad (7)$$

Note that for Type I stations $\mu_{im} = 0$ for $m \neq 1$, and that for Type IV stations $\mu_{im} = 0$ if the job being served is not in the m^{th} phase. The description of the other rates is as follows:

For Type I stations, we have $\mu_{i1}(\ldots) = \mu_1$. For Type II and III stations we have $\mu_{im}(T^{i}_{m+1,m}(\hat{k})) = \mu_{im}(k_i, k_{im}+1)$, $\mu_{im}(T^{ji}_{n,m}(\hat{k})) = \mu_{im}(k_i+1, k_{im}+1)$, $\mu_{im}(T^{jil}_{n,m,p}(\hat{k})) = \mu_{im}(k_i, k_{im}+1)$, $\mu_{im}(T^{.i,j.}_{m}(\hat{k})) = \mu_{im}(k_i+1, k_{im}+1)$. For Type IV stations, we have $\mu_{im}(T^{i}_{m+1,m}(\hat{k})) = \mu_{im}$ for $m = t_{i1}-1$, and $\mu_{im}(T^{ji}_{n,m}(\hat{k})) = \mu_{im}(T^{jil}_{n,m,p}(\hat{k})) = \mu_{im}(T^{.i,j.}_{m}(\hat{k})) = \mu_i$.

From the definition of the states $T^{i}_{m+1,m}(\hat{k})$, $T^{ji}_{n,m}(\hat{k})$, $T^{jil}_{n,m,p}(\hat{k})$, $T^{.i,j.}_{m}(\hat{k})$, and from the product form solution (6) we derive

$$P(T^{i}_{m+1,m}(\hat{k})) = P(\hat{k})\frac{A_{im}}{A_{im+1}}\frac{\mu_{im+1}(k_i, k_{im+1})}{\mu_{im}(k_i, k_{im}+1)}, \qquad (8)$$

$$P(T^{ji}_{n,m}(\hat{k})) = P(\hat{k})\frac{e_iA_{im}}{\mu_{im}(k_i+1, k_{im}+1)}\frac{\mu_{jn}(k_j, k_{jn})}{e_jA_{jn}}, \qquad (9)$$

$$P(T^{jil}_{n,m,p}(\hat{k})) = P(\hat{k})\frac{e_iA_{im}}{\mu_{im}(k_i, k_{im}+1)}\frac{\mu_{ip}(k_i, k_{ip})}{e_iA_{ip}}\frac{\mu_{jn}(k_j, k_{jn})}{e_jA_{jn}}\frac{e_lp_{li}}{\sum_{r=1}^{S_i}\frac{f_{ir}}{A_{ir}}\mu_{ir}(k_i, k_{ir})}, \qquad (10)$$

$$P(T^{.i,j.}_{m}(\hat{k})) = P(\hat{k})\frac{e_iA_{im}}{\mu_{im}(k_i+1, k_{im}+1)}\frac{\sum_{r=1}^{S_j}\frac{f_{jr}}{A_{jr}}\mu_{jr}(k_j, k_{jr})}{e_ip_{ij}}. \qquad (11)$$

We denote the first, second, third, and fourth terms of the right-hand side of the global balance equation (7) by T_1, T_2, T_3, and T_4, respectively. From equation (8) and expression (2) we derive

$$T_1 = \sum_{i=1}^{N}\sum_{m=1}^{S_1} P(\hat{k})\mu_{im+1}(k_i, k_{im+1})\frac{A_{im}}{A_{im+1}}a_{im}.$$

Using the fact that $\sum_{m=1}^{S_i} A_{im} b_{im} = 1$ and the traffic equation (5) we get

$$T_2 = \sum_{j \notin \Gamma} P(\hat{k}) \sum_{n=1}^{S_j} \frac{f_{jn}}{A_{jn}} \mu_{jn}(k_j, k_{jn}) - \sum_{j \notin \Gamma} \sum_{i \in \Gamma} P(\hat{k}) \frac{e_i p_{ij}}{e_j} \sum_{n=1}^{S_j} \frac{f_{jn}}{A_{jn}} \mu_{jn}(k_j, k_{jn}).$$

In a similar way, we obtain

$$T_3 = \sum_{j \notin \Gamma} \sum_{i \in \Gamma} P(\hat{k}) \frac{e_i p_{ij}}{e_j} \sum_{n=1}^{S_j} \frac{f_{jn}}{A_{jn}} \mu_{jn}(k_j, k_{jn}),$$

and

$$T_4 = \sum_{j \in \Gamma} P(\hat{k}) \sum_{n=1}^{S_j} \frac{f_{jn}}{A_{jn}} \mu_{jn}(k_j, k_{jn}).$$

So,

$$T_1 + T_2 + T_3 + T_4 = \sum_{j=1}^{N} \sum_{m=1}^{S_1} P(\hat{k}) \mu_{jm+1}(k_j, k_{jm+1}) \frac{A_{jm}}{A_{jm+1}} a_{jm} + \sum_{j=1}^{N} P(\hat{k}) \sum_{n=1}^{S_j} \frac{f_{jn}}{A_{jn}} \mu_{jn}(k_j, k_{jn}).$$

Since $\frac{A_{im}}{A_{im+1}} a_{im} + \frac{f_{im+1}}{A_{im+1}} = 1$, the global balance equation is verified, i.e.,

$$P(\hat{k}) \sum_{i=1}^{N} \sum_{m=1}^{S_i} \mu_{im}(k_i, k_{im}) = T_1 + T_2 + T_3 + T_4.$$

Q.E.D.

Note that in this proof condition (4) on f_{im} is necessary to avoid zero in the denominator of the product form solution although $\sum_{m=1}^{N} k_{im} = B_i$ which is sufficient in the exponential case.

4 Steady-State Distribution for Blocked and Unblocked Jobs

For each full station i, i.e., $k_i = B_i$, we define a multinomial distribution $M_i(k_i, (\frac{e_i A_{im}}{\mu_{im}})_{1 \leq m \leq S_i})$. We denote by E_i its expectation. For a positive and bounded function f we have

$$E_i[f(k_{i1}, \ldots, k_{iS_i})] = \sum_{k_{im} s.t. \sum_{m=1}^{S_i} k_{im} = k_i} k_i! \prod_{m=1}^{S_i} \frac{1}{k_{im}!} (\frac{e_i A_{im}}{\mu_{im}})^{k_{im}} f(k_{i1}, \ldots, k_{iS_i}).$$

With these notations we can state:

Theorem 2 *The steady-state distribution of the mean number of blocked and unblocked jobs is given by*

$$P((k_1, b_1), \ldots, (k_N, b_N)) = \gamma \prod_{i=1}^{N} G_i(k_i, b_i) \tag{12}$$

where

$$G_i(k_i, b_i) = \begin{cases} (e_i \mu_i^{-1})^{k_i} & \text{if } i \text{ is of Type I, II, or IV and } b_i{=}0 \\ \frac{1}{k_i!}(e_i \mu_i^{-1})^{k_i} & \text{if } i \text{ is of Type III and } b_i = 0 \\ E_i[\frac{e_i}{\sum_{m=1}^{S_i} \frac{f_{im}}{A_{im}} \mu_{im}(\bar{k}_i)}]^{b_i} & \text{if } i \text{ is of Type I, II, or IV and } b_i > 0 \\ \frac{1}{k_i!} E_i[\frac{e_i}{\sum_{m=1}^{S_i} \frac{f_{im}}{A_{im}} \mu_{im}(\bar{k}_i)}]^{b_i} & \text{if } i \text{ is of Type III and } b_i > 0 \end{cases}$$

Proof.

The first cases ($b_i = 0$) are as in the BCMP queueing networks. The other cases ($b_i > 0$) are obtained for a given number of unblocked jobs $(k_{im})_{1 \le m \le S_i}$, by considering only the number of jobs blocked by station i in the other stations without considering their positions, i.e.,

$$G_i((k_{im})_{1 \le m \le S_i}, (b_{ij})_{1 \le j \le N}) = \sum_{\bar{n}_i, b_{ij} \, fixed} F_i(\hat{k}).$$

The combinatorics theory gives

$$G_i((k_{im})_{1 \le m \le S_i}, (b_{ij})_{1 \le j \le N}) = \begin{cases} k_i! \prod_{m=1}^{S_i} \frac{(e_i A_{im} \mu_{im}^{-1})^{k_{im}}}{k_{im}!} b_i! \prod_{j=1}^{N} \frac{1}{b_{ij}!} (\frac{e_j p_{ji}}{\sum_{m=1}^{S_i} \frac{f_{im}}{A_{im}} \mu_{im}(k_i, k_{im})})^{b_{ij}} \\ \text{if } i \text{ is a Type I, II, or IV station} \\ \prod_{m=1}^{S_i} \frac{(e_i A_{im} \mu_{im}^{-1})^{k_{im}}}{k_{im}!} b_i! \prod_{j=1}^{N} \frac{1}{b_{ij}!} (\frac{e_j p_{ji}}{\sum_{m=1}^{S_i} \frac{f_{im}}{A_{im}} \mu_{im}(k_i, k_{im})})^{b_{ij}} \\ \text{if } i \text{ is a Type III station} \end{cases}$$

By summing the last expression over $(b_{ij})_{1 \le j \le N}$ with fixed b_i, i.e.,

$$H_i((k_{im})_{1 \le m \le S_i}, b_i)) = \sum_{(b_{ij})_{1 \le j \le N} s.t. \sum_{j=1}^{N} b_{ij} = b_i} G_i((k_{im})_{1 \le m \le S_i}, (b_{ij})_{1 \le j \le N}), \tag{13}$$

and according to Newton formula [15], we obtain

$$H_i((k_{im})_{1\le m\le S_i}, b_i)) = \begin{cases} k_i! \prod_{m=1}^{S_i} \frac{(e_i A_{im} \mu_{im}^{-1})^{k_{im}}}{k_{im}!} \left(\frac{e_i}{\sum_{m=1}^{S_i} \frac{f_{im}}{A_{im}} \mu_{im}(k_i, k_{im})}\right)^{b_i} \\ \text{if } i \text{ is a Type I, II, or IV station} \\ \prod_{m=1}^{S_i} \frac{(e_i A_{im} \mu_{im}^{-1})^{k_{im}}}{k_{im}!} \left(\frac{e_i}{\sum_{m=1}^{S_i} \frac{f_{im}}{A_{im}} \mu_{im}(k_i, k_{im})}\right)^{b_i} \\ \text{if } i \text{ is a Type III station} \end{cases}$$

The theorem follows from the definition of a multinomial distribution and by summing H_i over $(k_{im})_{1\le m\le S_i}$ with fixed $k_i = \sum_{m=1}^{S_i} k_{im}$.

Q.E.D.

5 Computation of the Normalization Constant

The normalization constant can be determined, according to Theorem (2) using only the steady-state distribution of the number of blocked and unblocked jobs in the network:

$$\frac{1}{\gamma} = \sum_{k_N=0}^{B_N} \sum_{b_N=0}^{K-k_N} \sum_{k_{N-1}=0}^{B_{N-1} \bigwedge (K-(k_N+b_N))} \sum_{b_{N-1}=0}^{K-(k_N+b_N+k_{N-1})} \sum \cdots$$
$$\cdots \sum_{k_i=0}^{B_i \bigwedge (K-(\sum_{j=1}^{i+1}(k_j+b_j)))} \sum_{b_i=0}^{K-(\sum_{j=1}^{i+1}(k_j+b_j))+k_i} \cdots \sum \cdots \prod_{j=1}^{N} G_j(k_j, b_j). \qquad (14)$$

In order to simplify the notations, we use (for $i = 1, \ldots, N$)

$$g_i(k_i) = \begin{cases} G_i(k_i, 0) & \text{if } k_i \le B_i, \\ G_i(B_i, k_i - B_i) & \text{otherwise.} \end{cases}$$

Equation (14) can then be rewritten as

$$\frac{1}{\gamma} = \sum_{k_N}^{K} \sum_{k_{N-1}=0}^{K-k_N} \cdots \sum_{k_i=0}^{K-\sum_{j=N}^{i+1} k_j} \cdots \sum_{k_1=K-\sum_{j=N}^{2} k_j} \prod_{l=1}^{N} g_l(k_l),$$
$$= \sum_{k_N}^{K} g_N(k_N) \sum_{k_{N-1}=0}^{K-k_N} \cdots \sum_{k_i=0}^{K-\sum_{j=N}^{i+1} k_j} \cdots \sum_{k_1=K-\sum_{j=N}^{2} k_j} \prod_{l=1}^{N-1} g_l(k_l),$$

$$= \sum_{k_N}^{K} g_N(k_N) C_{N-1}(K - k_N), \tag{15}$$

where $C_{N-1}(K - k_N)$ is a normalization constant associated with an appropiate queueing network $B(N-1, K-k_N)$. We deduce by recurrence that equation (15) may be expressed in a compact form as

$$\frac{1}{\gamma} = g_1 \otimes g_2 \otimes \ldots \otimes g_N(K), \tag{16}$$

where $\otimes$ is the convolution operator. Hereafter, a convolution algorithm is derived from equation (16) to compute the normalization constant γ. Let C_j denote the normalization constant of the first j stations.

For $j = 1$,

$$C_1(k) = g_1(k) \quad 0 \leq k \leq K \tag{17}$$

For $j = 2, \ldots, N$,

$$C_j(k) = \sum_{i=0}^{k} C_{j-1}(i) g_j(k-i) \quad 0 \leq k \leq K \tag{18}$$

So that

$$\frac{1}{\gamma} = C_N(K). \tag{19}$$

Similar to the convolution algorithm in the BCMP case, some simplifications are possible using the expressions of g_j. We assume that stations are numbered so that stations 1 to J are of Type III and stations $J+1$ to N are of Type I, II, or IV. Then, for $0 \leq k \leq min_{1 \leq j \leq J}(B_j)$ and $b_j = 0$ if $k_j = B_j$, we have

$$C_j(k) = \frac{1}{k!} [\sum_{i=1}^{j} e_i (\sum_{m=1}^{S_i} A_{im} \mu_{im}^{-1})]^k \qquad 1 \leq j \leq J. \tag{20}$$

For $0 \leq k \leq min_{max(2,J+1) \leq j \leq N}(B_j)$ and $b_j = 0$ if $k_j = B_j$, equation (18) simplifies to

$$C_j(k) = C_{j-1}(k) + C_j(k-1)(e_j \sum_{m=1}^{S_j} A_{jm} \mu_{jm}^{-1}) \qquad max(2, J+1) \leq j \leq N. \tag{21}$$

Equations (20) and (21) can be used as initial values of $C_j(k)$. For the the remaining terms, we are forced to use equation (18) as in the BCMP networks with load dependent parameters. For the exponential case, relation (21) holds for all $0 \leq k \leq N$.

In the next section we derive the steady state distributions of the number of blocked and unblocked jobs in terms of the normalization constant algorithm. Then we derive the new formulae for performance measures.

6 Performance Measures

We denote by $P(k_j)$ the steady-state distribution that there are k_j unblocked jobs in station j. According to product form solution (12) we have:

$$\begin{aligned} P(k_j) &= \sum_{k_i s.t. \sum_{i=1,i\neq j}^{N} k_i = K-k_j \& i\neq j} P(k_1,\ldots,k_j,\ldots,k_N) \\ &= \frac{1}{C_N(K)} g_j(k_j) \sum_{k_i s.t. \sum_{i=1,i\neq j}^{N} k_i = K-k_j \& i\neq j} \prod_{i=1,i\neq j}^{N} g_i(k_i) \\ &= \frac{C_{N-(j)}(K-k_j)}{C_N(K)} g_j(k_j) \end{aligned} \tag{22}$$

where $C_{N-(j)}(K-k_j)$ is the normalization constant for the network without the j^{th} station and with $K-k_j$ jobs. Let $P(b_{ij})$ be the steady-state distribution of the number of blocked jobs by the i^{th} station ($k_i = B_i$) in station j. $P(b_{ij})$ is obtained as in Theorem 2:

$$\begin{aligned} P(b_{ij}) &= \sum_{b_i=b_{ij}}^{K-B_i} \frac{C_{N-(i)}(K-B_i-b_i)}{C_N(K)} \frac{b_i!}{k_{ij}!(b_i-b_{ij})!} \\ & E_i[(\frac{\sum_{l=1,l\neq j}^{N} e_l p_{li}}{\sum_{m=1}^{S_i} \mu_{im}(k_i,k_{im})})^{b_i-b_{ij}} (\frac{e_j p_{ji}}{\sum_{m=1}^{S_i} \mu_{im}(k_i,k_{im})})^{b_{ij}}]. \end{aligned} \tag{23}$$

We denote by $P(k_{j1},\ldots,k_{jS_j})$ the steady-state distribution that there are k_{jm} unblocked jobs in station j in the m^{th} phase ($1 \le m \le S_j$). Similar to the calculations (22), we obtain

$$P(k_{j1},\ldots,k_{jS_j}) = \frac{C_{N-(j)}(K-k_j-b_j)}{C_N(K)} H_j(k_{j1},\ldots,k_{jS_j},0), \quad for \quad k_j = \sum_{m=1}^{S_j} k_{jm} \le B_j - 1 \tag{24}$$

$$P(k_{j1},\ldots,k_{jS_j}) = \sum_{b_j=0}^{N-k_j} \frac{C_{N-(j)}(K-k_j-b_j)}{C_N(K)} H_j(k_{j1},\ldots,k_{jS_j},b_j) \quad for \quad k_j = \sum_{m=1}^{S_j} k_{jm} = B_j \tag{25}$$

where H is defined by equation (13).

Note that $b_j = 0$ if $k_j \leq B_j - 1$. Let $P(k_{j1}, \ldots, k_{jS_j}, B_i)$ be the steady-state distribution that there are k_{jm} unblocked jobs in station j at the m^{th} phase ($1 \leq m \leq S_j$) and that station i is full. For $k_j = \sum_{m=1}^{S_j} k_{jm} \leq B_j - 1$ we have

$$P(k_{j1}, \ldots, k_{jS_j}, B_i) = \frac{H_j(k_{j1}, \ldots, k_{jS_j}, 0)}{C_N(K)} \sum_{b_i=0}^{K-k_j-B_i} g_i(B_i + b_i) C_{N-(i,j)}(K - k_j - B_i - b_i) \quad (26)$$

where $C_{N-(i,j)}$ is the normalization constant for the network without the i^{th} and the j^{th} stations. We denote by $P(k_{i1}, \ldots, k_{iS_i}, \sum_{m=1}^{S_i} k_{im} = B_i, n_{i,1} = j)$ the steady-state distribution that the i^{th} station is full with the unblocked state $(k_{i1}, \ldots, k_{iS_i})$ and that the first blocked job by station i is a job in station j. According to the Bernoulli expression of the number of blocked jobs distribution (6), the steady state distribution that the first blocked job by station i belongs to station j is

$$P(k_{i1}, \ldots, k_{iS_i}, \sum_{m=1}^{S_i} k_{im} = B_i, n_{i,1} = j) = P(k_{i1}, \ldots, k_{iS_i}, \sum_{m=1}^{S_i} k_{im} = B_i, b_i \geq 1) \frac{e_j p_{ji}}{e_i} \quad (27)$$

where $P(k_{i1}, \ldots, k_{iS_i}, \sum_{m=1}^{S_i} k_{im} = B_i, b_i \geq 1)$ is the distribution that station i is full and that there is more than one job blocked by this station. This distribution is given by:

$$P(k_{i1}, \ldots, k_{iS_i}, \sum_{m=1}^{S_i} k_{im} = B_i, b_i \geq 1) = \sum_{b_i=1}^{N-B_i} H_i(k_{i1}, \ldots, k_{iS_i}, b_i) C_{N-(i)}(K - B_i - b_i). \quad (28)$$

Theorem 3 *The throughput of the j^{th} station is computed from*

$$\begin{aligned} \lambda_j = & \sum_{k_{j1}, \ldots, k_{jS_j}} P(k_{j1}, \ldots, k_{jS_j}) \sum_{m=1}^{S_j} \mu_{jm}(k_j, k_{jm}) \\ & - \sum_{i=1}^{N} P(k_{j1}, \ldots, k_{jS_j}, B_i) \sum_{m=1}^{S_j} \mu_{jm}(k_j, k_{jm}) p_{ji} \\ & + \sum_{i=1}^{N} P(k_{i1}, \ldots, k_{iS_i}, \sum_{m=1}^{S_i} k_{im} = B_i, n_{i,1} = j) \sum_{m=1}^{S_i} \mu_{im}(k_i, k_{im}). \end{aligned}$$

where $P(k_{j1}, \ldots, k_{jS_j})$, $P(k_{j1}, \ldots, k_{jS_j}, B_i)$ and $P(k_{i1}, \ldots, k_{iS_i}, \sum_{m=1}^{S_i} k_{im} = B_i, n_{i,1} = j)$ are given by equations (24), (25), (26), (27) and (28), respectively.

The mean number of jobs in the j^{th} station is given by

$$k_j = \sum_{k_j=1}^{B_j} k_j P(k_j) + \sum_{i=1}^{N} \sum_{b_{ij}=1}^{K-B_i} b_{ij} P(b_{ij})$$

where $P(k_j)$ and $P(b_{ij})$ are obtained from equations (22) and (23), respectively.

Proof.

A transition occurs from station j to station i when an active job in station j finishes its service and is routed to station i, which must not be full. A second type of transition may occur also when a blocked job in station j by station i is in the first position and a transition occurs in station i. Consequently, the blocked job in station j is moved to station i.

$$\lambda_j = \sum_{\hat{k}} P(\hat{k})[\sum_{m=1}^{S_j} \mu_{jm}(k_j, k_{jm})(1 - \sum_{i=1}^{N} I_{k_i=B_i} p_{ji}) + \sum_{i=1}^{N}\sum_{m=1}^{N} \mu_{im}(k_i, k_{im}) I_{k_i=B_i} I_{n_{i,1}=j}]$$

where I_E is the indicator of event E and is equal to 1 if E is true and zero otherwise. The first assertion of the Theorem is obtained from the last equation.
The mean number of jobs in station j is equal to the mean number of unblocked jobs plus the mean number of jobs blocked in station j by other stations.

Q.E.D.

7 Conclusions

In this paper we obtained a product form solution for a closed queueing network model with finite buffer capacities. The stations can be either of Type I, II, III, or IV with finite buffers. The blocking mechanism is the kanban blocking mechanism. The routing matrix is assumed such that two full stations do not interact. This condition is required to avoid the move of two jobs at the same time. The service time distributions are assumed to be exponential for Type I stations and Coxian for Type II, III and IV stations. Each phase of the Coxian representation may be reached with a positive probability in the beginning of the service. The number of blocked and unblocked jobs steady-state distribution depends on the Cox phase rates. Thus, the insensitivity property does not hold here in contrast to the classical BCMP network model. We developed a convolution algoritm for the computation of the normalization constant. We derived new formulae for the performance measures.

REFERENCES

[1] I. F. Akyildiz, "Exact Product Form Solution for Queueing Networks with Blocking", *IEEE Transactions on Computers*, Vol. **C-36**, No. 1, January 1987, pp. 122-125.

[2] I. F. Akyildiz and H. G. Perros. "Special Issue on Queueing Networks with Finite Capacity Queues: Introduction". *Performance Evaluation*, 10(3), Dec. 1989.

[3] I. F. Akyildiz and H. von Brand, "Exact Solutions for Networks of Queues with Blocking-After-Service", *Technical Report, Georgia Tech, College of Computing, April 1991, GIT-COC-91-26.*

[4] I.F. Akyildiz and C.C. Huang, "Exact Analysis of Queueing Networks with Multiple job Classes and Blocking-After-Service", to appear in *Queueing Systems: Theory and Applications: QUESTA*, 1993.

[5] I. F. Akyildiz and J. Liebeherr, "Optimal Deadlock Free Buffer Allocation in Multiple Chain Blocking Networks of Queues", *Proc. of Performance of Distributed Systems*, North-Holland, September 1991, pp. 245-264.

[6] F. Baskett, K. M. Chandy, R. R. Muntz and F. G. Palacios, "Open, Closed and Mixed Networks of Queues with Different Classes of Customers", *Journal of the ACM*, Vol. **22**, Vol. 15, pp. 248-260, April 1975.

[7] F. P. Kelly, "Networks of Queues with Customers of Different Types", *Journal of Applied Probability*, Vol. **12**, pp. 542-554, 1975.

[8] F. P. Kelly, "Networks of Queues", *Advances in Applied Probability*, Vol. **8**, No. 2, pp. 416-432, June 1976.

[9] D. Mitra and I. Mitrani, "Analysis of a Kanban Discipline for Cell Coordination in Production Lines: Part I", *Management Science*, Vol. 36, 1990, pp. 1548-1566.

[10] D. Mitra and I. Mitrani, "Analysis of a Kanban Discipline for Cell Coordination in Production Lines: Part II: Stochastic Demands", *Operations Research*, Vol. 39, No. 5, Sept-Oct. 1991, pp. 807-824.

[11] Y. Monden, "Adaptable Kanban System Helps Toyota Maintain Just-in-Time Production", *Industrial Engineering*, May 1981, pp. 29-46.

[12] R. O. Onvural. "A Survey of Closed Queueing Networks with Finite Buffers". *ACM Computing Surveys*, 22(2):83–121, June 1990.

[13] R. O. Onvural and H. G. Perros. "Equivalencies between Closed Queueing Networks with Finite Buffers". *Performance Evaluation*, 9:263–269, 1988/89.

[14] R. O. Onvural and I. F. Akyildiz. "Queueing Networks with Finite Buffer Capacities". *Proceedings of the Workshop on Queueing Networks with Blocking*, (editors), North-Holland Publ. Comp., December 1992.

[15] M. Reiser and H. Kobayashi, "Queueing Networks with Multiple Closed Chains: Theory and COmputational Algorithms", *IBM Journal Research Dev.*, No. 19, May 1975, pp. 283-294.

[16] M. Sepehri, "How Kanban System is Used in an American Toyota Motor Facility", *Industrial Engineering*, Febr. 1985, pp. 50-56.

Deadlock Freedom in Queueing Networks with Multiple Routing Chains and Finite Capacities *

Ian F. Akyildiz † *Jörg Liebeherr* ‡

† School of Electrical Engineering,
Georgia Institute of Technology,
Atlanta, GA 30332,
U. S. A.

‡ Computer Science Department,
University of Virginia,
Charlottesville, VA 22903,
U. S. A.

Abstract

Blocking can lead to a deadlock state due to the finite buffer capacities in queueing network models. In this paper deadlock freedom is investigated in multiple chain queueing networks with blocking, and necessary and sufficient conditions are given. An algorithm is presented for finding a deadlock free buffer allocation with the least number of buffers, thus the optimal allocation. The algorithm maps the queueing network into a directed graph. A procedure for finding a subset of all cycles in the directed graph is used to generate the conditions for deadlock freedom of the network. Finally, the optimal buffer allocation is obtained from these conditions by applying linear programming techniques. It is shown that the effort to find an optimal buffer allocation can be reduced by taking advantage of structural properties of the network.

*This work was supported by the National Science Foundation under Grant No. CCR-90-11981

1 Introduction

Queueing networks have increased their importance in performance evaluation of computer systems and communication networks in the last two decades. Numerous methods have been developed for the analysis of systems with infinite station capacities. Since in actual systems the resources have finite capacities, queueing networks with blocking must be used for performance analysis. Each station of a queueing network with blocking can be thought of as a device with a finite buffer length. The network is simply a set of arbitrarily linked stations. Blocking arises due to the limitations imposed by the capacity of these stations. In particular, blocking occurs when the flow of jobs through one station may be stopped for a moment if a destination station has reached its full capacity. The set of rules that dictate when a station becomes blocked or unblocked is commonly referred as the blocking mechanism. There are few blocking mechanisms that have been studied in the literature [1].

In this work we consider the so-called blocking-after-service (BAS), also referred as type 1 or manufacturing blocking mechanism. A job upon service completion at a station i attempts to enter the destination station j. If station j is full at that moment, the job is forced to wait in station i's server until it can enter destination station j. The server remains blocked for this period of time. It cannot serve other jobs waiting in the queue.

Finite station capacities and blocking can introduce a deadlock situation. In a simple example, a deadlock may occur if a job which has finished its service at station i's server wants to join station j, whose capacity is full. That job is blocked in station i. Another job which has finished service at station j now wants to proceed to station i, whose capacity is also full. It blocks the jth station. Both jobs are waiting for each other. As a result a deadlock situation arises. There are two possible solutions for the deadlock problem: Either include a strategy to handle deadlocks in the model or simply restrict yourself to cases where deadlock is impossible. We select the second solution and find conditions under which a multiple chain blocking network is deadlock free. Kundu and Akyildiz [2] showed that queueing networks with a single routing chain are deadlock free if the number of jobs in the network is less than the capacity of the directed cycle with minimal capacity. However, the conditions for deadlock freedom with multiple routing chains become very complex due to interactions between routing chains.

We demonstrate these interactions in the network model given in Figure 1. There are 2 stations

and 2 routing chains in the model. Let B_{ir} denote the buffer of chain r at station i (for $i, r = 1, 2$) and let K_r denote the total number of jobs in chain r ($r = 1, 2$). Let the parameters be given by:

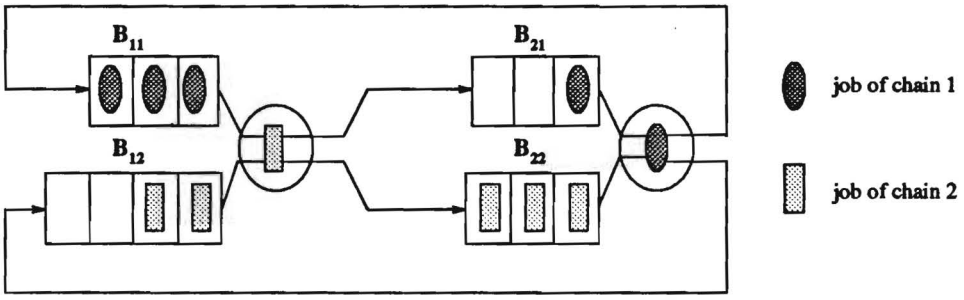


Figure 1: Blocking Network with 2 chains.

$K_1 = 5$; $K_2 = 6$

Capacity of $B_{11} = 3$; Capacity of $B_{21} = 3$

Capacity of $B_{12} = 4$; Capacity of $B_{22} = 3$

Here a deadlock situation occurs, even though the condition for deadlock freedom given in [2] are satisfied for each routing chain in isolation. In addition to determining the conditions for deadlock freedom it is an important issue find an allocation of station capacities in a queueing network such that deadlocks cannot occur.

In this study, we provide the necessary and sufficient conditions for a deadlock free blocking network with multiple routing chains. In section 2 we describe the class of models which is considered in this study. The conditions for deadlock freedom are stated and proved in section 3. In section 4 we give a definition of an optimal buffer allocation. Then we present an optimization algorithm which generates an optimal buffer allocation. Since the algorithm is not restricted to certain topologies, it generalizes the algorithm in [2] for the case of single chain networks. A crucial element of the buffer allocation algorithm is an efficient method to identify cycles in a directed graph. This method is discussed in section 5. In section 6 we show how to reduce the effort to find an optimal buffer allocation for networks with special topologies. In section 7, we show applications of the allocation algorithm. We summarize our results in section 8.

2 Model Description

We consider a closed queueing network Γ with the following properties:

a) The system consists of N stations and R disjoint routing chains. Stations and routing chains are referred to by their indexes. Let $\mathcal{N}$ and $\mathcal{R}$ denote the set of stations and routing chains, respectively. Let $\mathcal{R}_i \subseteq \mathcal{R}$ denote the set of routing chains visiting station i. Let $\mathcal{N}_r \subseteq \mathcal{N}$ denote the set of stations visited by routing chain r.

b) Each station i has a single server. The service time distribution and the scheduling discipline of a station is arbitrary, but non-preemptive.

c) Each station keeps separate buffers for jobs from different routing chains. B_{ir} denotes the buffer at station i (excluding the server) for jobs from routing chain r ($i \in \mathcal{N}$, $r \in \mathcal{R}_i$). Each buffer may accommodate only a finite number of jobs. Let Φ be an assignment of capacities to the buffers of Γ, i.e.,

$$\Phi : \{B_{ir} | i \in \mathcal{N},\ r \in \mathcal{R}_i\} \rightarrow \{0, 1, 2, \ldots\ldots\} \tag{1}$$

Note that a buffer can have infinite capacity ($\Phi(B_{ir}) = \infty$) or no capacity at all ($\Phi(B_{ir}) = 0$).

d) The number of jobs in the system is fixed at

$$\underline{K} = (K_1, K_2, \ldots, K_R) \tag{2}$$

where

$$K_r = \sum_{i \in \mathcal{N}_r} k_{ir}, \qquad \text{for } r = 1, \ldots, R \tag{3}$$

represents the total number of jobs in routing chain r, and k_{ir} is the number of jobs of chain r in the i-th station (in buffer and server). The total number of jobs in the network is denoted by $K = \sum_{r=1}^{R} K_r$. The state of the network is represented by a vector $\underline{k} = (\underline{k}_1, \underline{k}_2, \ldots, \underline{k}_N)$ where $\underline{k}_i$ is a vector $(k_{ir_1}, k_{ir_2}, \ldots, k_{ir_{|\mathcal{R}_i|}})$ with $r_u \in \mathcal{R}_i$ ($u = 1, 2, \ldots, |\mathcal{R}_i|$). The total capacity of station i is computed by $\sum_{r \in \mathcal{R}_i} \Phi(B_{ir}) + 1$.

e) A job of routing chain r which has received service by station i proceeds to the station j with probability $p_{ij,r}$ (for $i, j \in \mathcal{N}$, $r \in \mathcal{R}_i$ and $r \in \mathcal{R}_j$). The number of jobs in the buffer of station

j for jobs of chain r cannot exceed its capacity $\Phi(B_{jr})$. If B_{jr} is *saturated*, i.e., $k_{jr} = \Phi(B_{jr})$, the job of chain r is blocked at buffer B_{jr} and has to remain in the server of station i until a place in B_{jr} becomes available. This is the BAS (blocking-after-service) blocking mechanism.

3 Conditions For Deadlock Freedom

In this section we present the deadlock freedom conditions for a queueing network as described in section 2. Before we state the theorem, we need the following definitions:

Definition 1 *A buffer cycle (of length $t-1$ ($1 \leq t-1 \leq |\mathcal{N}|$)) is a sequence of buffers $C = (B_{\sigma_1\rho_1}, B_{\sigma_2\rho_2}, \ldots, B_{\sigma_{t-1}\rho_{t-1}}, B_{\sigma_t\rho_t})$ such that*

$$\begin{aligned} &\sigma_u \in \mathcal{N} && (1 \leq u \leq t) \\ &\sigma_t = \sigma_1 && \\ &\rho_u \in R_{\sigma_u},\ \rho_{u+1} \in R_{\sigma_u},\ p_{\sigma_u\sigma_{u+1},\rho_{u+1}} > 0 && (1 \leq u < t). \end{aligned} \tag{4}$$

Let $\mathcal{C}$ denote the set of all buffer cycles in Γ.

Definition 2 *The set of stations with buffers from routing chain r in buffer cycle C is defined by*

$$S_r^{(C)} = \{i \mid B_{ir} \in C, i \in \mathcal{N}, r \in \mathcal{R}_i\} \tag{5}$$

$S_r^{(C)}$ denotes the maximum number of servers attached to the buffers in cycle C which can be blocked by jobs from chain r. For example, $C_1 = (B_{11}, B_{21}, B_{11})$ and $C_2 = (B_{11}, B_{22}, B_{11})$ are two buffer cycles in the network shown in Figure 1. From Definition 2 we obtain $S_1^{(C_1)} = \{1,2\}$, $S_2^{(C_1)} = \emptyset$, $S_1^{(C_2)} = \{1\}$, and $S_2^{(C_2)} = \{2\}$. Next we state the conditions for deadlock freedom.

Theorem 1 *A multiple chain queueing network Γ is* deadlock free *(DLF) if and only if for all $C \in \mathcal{C}$ with $C = (B_{\sigma_1\rho_1}, B_{\sigma_2\rho_2}, \ldots, B_{\sigma_{t-1}\rho_{t-1}}, B_{\sigma_t\rho_t})$ there exists a routing chain $r \in \mathcal{R}$ such that:*

$$\sum_{(1<u\leq t)\wedge(\sigma_u \in S_r^{(C)})} \Phi(B_{\sigma_u r}) + |S_r^{(C)}| > K_r \tag{6}$$

Therefore, in each buffer cycle C there must be at least one routing chain r which, at the same time, cannot saturate all its buffers in cycle C and have the maximum number of jobs $S_r^{(C)}$ jobs blocked.

Proof:

1. *Necessity:*

   Assume there exists a cycle $\hat{C} = (B_{\sigma_1\rho_1}, B_{\sigma_2\rho_2}, \ldots, B_{\sigma_{i-1}\rho_{i-1}}, B_{\sigma_i\rho_i})$ with:

$$(\forall r \in \mathcal{R}) \; (\sum_{1<u\leq\hat{\imath}\wedge\sigma_u\in S_r^{(\hat{C})}} \Phi(B_{\sigma_u r}) + |S_r^{(C)}| \leq K_r). \tag{7}$$

   Then, a state is feasible where all buffers in $\hat{C}$ are saturated, i.e., $B_{\sigma_u\rho_u}$ contains $\Phi(B_{\sigma_u\rho_u})$ jobs of chain ρ_u $(1 < u \leq \hat{\imath})$. Additionally, the server of station σ_u may contain a job of chain ρ_{u+1}. Note that the entire network will have $|S_r^{(C)}|$ servers which contain a job of chain r $(r \in \mathcal{R})$. There exists a positive probability that the job in station σ_u's server has picked station σ_{u+1} as its next destination. In this state, no server can release a job and eventually, each station σ_u is blocked. Thus, a deadlock persists. □

2. *Sufficiency:*

   Assume the multiple chain queueing network with allocation Φ is in a deadlock state. Then there must exist a permanently blocked station, say σ_1, which holds a job from routing chain, say ρ_2 $(\rho_2 \in \mathcal{R}_{\sigma_1})$, in its server. The job is blocked at a saturated buffer, say $B_{\sigma_2\rho_2}$ of some station, say σ_2, i.e., $k_{\sigma_2\rho_2} = \Phi(B_{\sigma_2\rho_2})$. Station σ_2 itself must be blocked, otherwise a space in $B_{\sigma_2\rho_2}$ will eventually become available and station σ_1 would not be permanently blocked. The job in the server of station σ_2 may be blocked at a saturated buffer, say $B_{\sigma_3\rho_3}$ of some station, say σ_3, i.e., $k_{\sigma_3\rho_3} = \Phi(B_{\sigma_3\rho_3})$. We can continue to apply this argument. Because of the finiteness of stations and buffers, we will encounter a station σ_{t-1} which is blocked at a buffer of station σ_1, say $B_{\sigma_1\rho_1}$. Then, buffers $B_{\sigma_2\rho_2}, B_{\sigma_3\rho_3}, \ldots, B_{\sigma_1\rho_1}$ define a buffer cycle $\hat{C} = (B_{\sigma_1\rho_1}, B_{\sigma_2\rho_2}, \ldots, B_{\sigma_{i-1}\rho_{i-1}}, B_{\sigma_i\rho_i})$ $(B_{\sigma_1\rho_1} = B_{\sigma_i\rho_i})$. From $\hat{C}$ we construct the sets $S_r^{(\hat{C})}$. Note that a job from chain r is blocked in the server of station σ_{u-1} if $B_{\sigma_u\rho_u}$ is a buffer in the cycle. Therefore, $|S_r^{(\hat{C})}|$ jobs of chain r are blocked in servers of stations which have a buffer in the cycle. If we sum up the number of jobs which are in buffers and servers of the cycle for each chain r, we obtain for each chain:

$$\sum_{(1<u\leq\hat{\imath})\wedge(\sigma_u\in S_r^{(\hat{C})})} \Phi(B_{\sigma_u r}) + |S_r^{(\hat{C})}| \tag{8}$$

The sum of each chain r is less or equal the total number of jobs K_r. Therefore:

$$\sum_{(1<u\leq\hat{t})\wedge(\sigma_u\in S_r^{(\widehat{C})})} \Phi(B_{\sigma_u r}) + |S_r^{(\widehat{C})}| < K_r \tag{9}$$

However, this is a violation of the DLF condition (equation (6)) for cycle $\widehat{C}$. □

The following Corollary follows directly from Theorem 1:

Corollary 1 *The DLF condition is satisfied for a cycle $C \in \mathcal{C}$, if there exists an $r \in \mathcal{R}$ with:*

$$|S_r^{(C)}| > K_r \tag{10}$$

In other words, if the number of stations with buffers from a particular routing chain in a cycle exceeds the total number of jobs of that chain, then the deadlock freedom condition for that cycle is satisfied disregarding the capacities of the buffers.

If the DLF conditions (equation (6)) are satisfied for a buffer cycle C, they are clearly satisfied for all cycles which include the buffers of cycle C as a subset. Therefore, it is not required to test the DLF conditions for all buffer cycles in order to guarantee deadlock freedom of the network. This is formalized in the following Lemma:

Lemma 1 *Given a multiple chain queueing network Γ and the set of all buffer cycles $\mathcal{C}$ of Γ. For any $C', C'' \in \mathcal{C}$ let $C' \subseteq C''$, if the set of buffers in C' is a subset of the set of buffers in C''. Let $\tilde{\mathcal{C}} \subseteq \mathcal{C}$ be such that*

$$(\forall C', C'' \in \tilde{\mathcal{C}})(C' \not\subseteq C''). \tag{11}$$

Then, Γ is deadlock free if the DLF condition (equation (6)) is satisfied for all $C \in \tilde{\mathcal{C}}$.

Proof: If $C' \subseteq C''$, then $S_r^{(C')} \subseteq S_r^{(C'')}$ for all $r \in \mathcal{R}$. Therefore, if the DLF condition is satisfied for a routing chain r in C', i.e., $\sum_{(1<u\leq t)\wedge(\sigma_u\in S_r^{(C')})} \Phi(B_{\sigma_u r}) + |S_r^{(C)}| > K_r$, it clearly is satisfied for any superset of C'. □

4 Deadlock Free Buffer Allocation Algorithm

Once the conditions for a deadlock free network are known, we can investigate the problem of finding a deadlock free allocation with the least number of buffers.

Definition 3 *A buffer allocation Φ^* for a queueing network Γ is optimal if:*

(i) Φ^* *satisfies the DLF condition (equation (6)) for all* $C \in \mathcal{C}$,

(ii) Φ^* *is minimal, i.e., no deadlock free allocation* Φ *for* Γ *allocates less total capacity to the buffers of* Γ *than* Φ^**:*

$$\sum_{i \in \mathcal{N}} \sum_{r \in \mathcal{R}_i} \Phi^*(B_{ir}) \leq \sum_{i \in \mathcal{N}} \sum_{r \in \mathcal{R}_i} \Phi(B_{ir}) \tag{12}$$

In the remaining part of this section we will present an algorithm which finds an optimal buffer allocation for a queueing network Γ. The algorithm is executed in two steps:

1. Find the set $\tilde{\mathcal{C}}$ of buffer cycles in Γ and establish the sets $S_r^{(C)}$ $(r \in \mathcal{R})$ for each $C \in \tilde{\mathcal{C}}$.
2. Obtain an integer program from $S_r^{(C)}$ $(r \in \mathcal{R}, C \in \tilde{\mathcal{C}})$. Solve the integer program with respect to the objective that the total number of allocated buffers be minimal.

In the following subsections we will discuss these steps in detail.

4.1 Finding Buffer Cycles

We approach the problem of finding the buffer cycles of a queueing network as the problem of finding the so-called elementary cycles in a directed graph. The following Lemma allows to map the queueing network into a directed graph:

Lemma 2 *Given a multiple chain queueing network* Γ.
Obtain a directed graph $G_\Gamma = (V_\Gamma, E_\Gamma)$ *with the set of vertices* V_Γ *and the set of arcs* E_Γ *from* Γ *by:*

(i) $V_\Gamma = \{B_{ir} |\ i \in \mathcal{N},\ r \in \mathcal{R}_i\}$,

(ii) $E_\Gamma = \{(B_{ir}, B_{js}) |\ r \in \mathcal{R}_i \ \wedge\ (\exists s \in \mathcal{R})(s \in \mathcal{R}_i \ \wedge\ s \in \mathcal{R}_j \ \wedge\ p_{ij,s} > 0)\}$.

Then, C is a buffer cycle in Γ if and only if C is a cycle in G_Γ.

Proof: The proof follows immediately from the definition of a buffer cycle and the construction of the directed graph. □

To obtain the list of cycles in a directed graph, we apply an algorithm which is based on a method to find elementary cycles in a directed graph [3]. The algorithm in [3], referred to as

Johnson's algorithm, generates a complete list of elementary cycles. However, we are not interested in all cycles of G_Γ. Rather, motivated by Lemma 1, we want to find the set of cycles $\tilde{\mathcal{C}}$ where no two cycles are a subset of each other. This restriction allows to improve the cycle finding algorithm significantly, compared to the original version of *Johnson's algorithm*. We discuss the details of the algorithm in section 5.

Corollary 1 states that the DLF condition (equation (6)) is satisfied for certain buffer cycles disregarding the capacities which are allocated to the buffers of the cycle. Therefore, buffer cycles which satisfy the condition in Corollary 1 (equation (10)) can be eliminated in the forthcoming steps of the algorithm. Note however, that the elimination is dependent on the total number of jobs in each chain K_r.

4.2 Optimization

The algorithm presented in the previous subsection provides the set of buffer cycles $\tilde{\mathcal{C}}$. These cycles need to be examined for deadlock freedom of an allocation. If the deadlock condition is satisfied for all buffer cycles in $\tilde{\mathcal{C}}$, then the network is deadlock free. We present two approaches for buffer allocation in a given network. The first approach is based on integer programming techniques and guarantees an optimal buffer allocation. The second approach is a heuristic method which always provides a deadlock free buffer allocation, but may yield a suboptimal solution. The advantage of the heuristic method is that it is computationally less demanding than solving the integer program.

4.2.1 Optimization with Integer Programming

Finding an optimal buffer allocation Φ^* for a given network Γ is the solution of the following optimization problem:

$$\begin{aligned} &\text{Find } \Phi^* \text{ which minimizes} \sum_{i\in\mathcal{N}} \sum_{r\in\mathcal{R}_i} \Phi^*(B_{ir}) \text{ with the constraints:} \\ &(\forall C \in \tilde{\mathcal{C}})(\exists r \in \mathcal{R})\ (C = (B_{\sigma_1\rho_1}, B_{\sigma_2\rho_2}, \ldots, B_{\sigma_{t-1}\rho_{t-1}}, B_{\sigma_t\rho_t}) \\ &\qquad \rightarrow \sum_{(1<u\leq t)\wedge(\sigma_u\in S_r^{(C)})} \Phi(B_{\sigma_u r}) + |S_r^{(C)}| > K_r) \end{aligned} \tag{13}$$

Let the elements of $\tilde{\mathcal{C}}$ be ordered, i.e., $\tilde{\mathcal{C}} = \{C_1, C_2, \ldots, C_{|\tilde{\mathcal{C}}|}\}$, and let t_l denote the number of buffers in cycle C_l. In the following, we denote $\Phi(B_{ir})$ by b_{ir} in order to stress that the allocation

of capacities to a buffer is a variable for the optimization process. Then, the optimization problem can be written as:

$$\begin{aligned}
&\textbf{minimize} \quad \sum_{i\in\mathcal{N}} \sum_{r\in\mathcal{R}_i} \mathrm{b}_{ir} \\
&\textbf{subject to} \\
&\qquad \bigvee_{S_r^{(C_l)}\neq\emptyset} \sum_{(1<u\leq t_l)\wedge(\sigma_u\in S_r^{(C_l)})} \mathrm{b}_{\sigma_u r} + |S_r^{(C_l)}| > K_r \quad (l = 1,2,\ldots,|\tilde{\mathcal{C}}|) \\
&\qquad \mathrm{b}_{ir} \in \{0,1,2,\ldots\} \qquad\qquad (i\in\mathcal{N},\ r\in\mathcal{R}_i).
\end{aligned} \tag{14}$$

By introducing additional $(0,1)$-variables which allow to replace the disjunctions in the constraints, we can state the optimization problem as an integer program:

$$\begin{aligned}
&\textbf{minimize} \quad \sum_{i\in\mathcal{N}} \sum_{r\in\mathcal{R}_i} \mathrm{b}_{ir} \\
&\textbf{subject to} \\
&\qquad \sum_{(1<u\leq t_l)\wedge(\sigma_u\in S_r^{(C_l)})} (\mathrm{b}_{\sigma_u r}) \;-\; \mathrm{y}_{lr}\cdot(K_r - S_r^{(C_l)} + 1) \;\geq 0 \\
&\qquad\qquad\qquad\qquad\qquad\qquad (l = 1,2,\ldots,|\tilde{\mathcal{C}}|;\ S_r^{(C_l)}\neq\emptyset) \\
&\qquad \sum_{S_r^{(C_l)}\neq\emptyset} \mathrm{y}_{lr} \geq 1 \qquad\qquad (l = 1,2,\ldots,|\tilde{\mathcal{C}}|) \\
&\qquad \mathrm{b}_{ir} \in \{0,1,2,\ldots\} \qquad\qquad (i\in\mathcal{N},\ r\in\mathcal{R}_i) \\
&\qquad \mathrm{y}_{lr} \in \{0,1\} \qquad\qquad (l = 1,2,\ldots,|\tilde{\mathcal{C}}|;\ S_r^{(C_l)}\neq\emptyset)
\end{aligned} \tag{15}$$

Note that variables y_{lr} are not needed for cycles which contain buffers from only one routing chain, i.e.,

$$(\exists r\in\mathcal{R})(\forall r'\in\mathcal{R})(r\neq r' \rightarrow S_r^{(C_l)} = \emptyset). \tag{16}$$

The optimization of the equation system in (15) can be solved with any program package for integer linear programs. The solution of the integer program provides values for b_{ir} which are an optimal buffer allocation Φ^* for network Γ. Note that in general, the system will have more than one optimal solution, since an optimization problem with the given structure will show considerable degeneracy, i.e., the system has multiple solutions which yield the same value for the objective function.

REMARK: Since the number of integer variables critically effects the computational complexity of the solution, the problem may become intractable if the size of the network is large. We propose

the following strategy: Relax the integrality constraint for variables b_{ir} and solve the problem as a mixed-integer linear program. Although the solution of the linear program in general does not necessarily provide integral solutions for b_{ir}, solving for b_{ir} as unrestricted variables will often provide integral solutions. If some b_{ir} are not integers, we add integrality constraints to the variables which yielded a non-integer value and re-iterate the solution. Eventually, all b_{ir} will have integer solutions. Note that this approach does not have any impact on the optimality of the final solution. We applied this strategy for all examples in section 7.

4.2.2 Heuristic Approach

The number of constraints and variables for the optimization process may become too large for a linear optimization technique. As an alternative, we present an approximate method for allocating buffer capacities. It is based on the heuristic that the buffer which appears in the most buffer cycles will have a non-zero capacity allocation. The method consecutively assigns capacities to the buffer b_{ir} which eliminates the most DLF conditions with the least buffer capacity. Since all DLF conditions are satisfied at the end of the procedure, we obtain a deadlock free allocation. However, there is no guarantee that the solution is optimal. To evaluate the quality of the approximation, we will present the approximate results when we discuss examples of buffer allocations for different queueing networks in section 7. We will see, that the approximation yields very accurate results, often identical with the optimal solution obtained by solving the integer program. The heuristic algorithm is executed as follows:

1.	Define $\mathcal{C}_{ir} = \{C \in \tilde{\mathcal{C}} \mid B_{ir} \in C\}$ $(i \in \mathcal{N},\ r \in \mathcal{R}_i)$.
2.	Define $b_{ir}^{(min)} = \min\{K_r - S_r^{(C)} + 1 \mid C \in \mathcal{C}_{ir},\ S_r^{(C)} \neq \emptyset\}$
3.	Select (i', r') with maximal ratio $E_{ir} = \dfrac{\lvert\mathcal{C}_{ir}\rvert}{b_{ir}^{(min)}}$.
4.	Assign $b_{i'r'} := b_{i'r'}^{(min)}$ and $\tilde{\mathcal{C}} := \tilde{\mathcal{C}} \setminus \mathcal{C}_{i'r'}$.
5.	If $\tilde{\mathcal{C}} \neq \emptyset$ GOTO 1.
6.	STOP.

5 Cycle Finding Algorithm

5.1 Algorithm

Here we discuss the algorithm which finds the set of elementary cycles $\tilde{C}$ [1] in a directed graph with the constraint that no two cycles are a subset of each other. The following terminology is needed for the discussion:

Definition 4 *Given a directed graph $G = (V, E)$.*

- *A cycle in G is elementary, if no vertex but the first and the last appear twice.*
- *G is strongly connected, if for any two vertices u and v there is a path from u to v.*
- *A strongly connected component (SCC) of a directed graph G is maximal strongly connected subgraph.*
- *The adjacency list $A_G(u)$ $(u \in V)$ of G contains v if $(u, v) \in E$. The adjacency structure A_G is the list of all adjacency lists $A_G(u)$ for $u \in V$.*

For our purposes we extend *Johnson's algorithm* (see section 4.1) to achieve a better performance for the cycle finding procedure. A simple extension is a fast search for Single-Vertex and Two-Vertex Cycles, i.e., cycles which involve only one vertex or two vertices:

- Single-Vertex cycles are not considered in [3]. However, they do frequently occur in a graph obtained from a queueing network. Therefore, we look for single vertex cycles and eliminates the vertex v from G if a cycle (v, v) exists:

$$\begin{aligned} V &:= V \setminus \{v\} \\ A_G &:= A_G \setminus \{A_G(v)\} \\ A_G(w) &:= A_G(w) \setminus \{v\} \quad (\text{for } w \in V_G). \end{aligned} \tag{17}$$

- A Two-Vertex cycle, i.e., a cycle (v, w, v) $(v, w \in V)$, can be easily identified from the adjacency structure. Assuming that vertices with a Single-Vertex cycle have been removed from the graph, any cycle along an arc (v, w) or (w, v) is a superset of a Two-Vertex cycle (v, w, v).

[1] Since buffer cycles in Γ can be mapped into cycles of graph G_Γ (Lemma 2), we use $\mathcal{C}$ to denote the set of all elementary cycles in a directed graph.

Since Lemma 1 allows to us disregard cycles that are contained in other cycles DLF conditions, we eliminate the edges (v, w) and (w, v) from the graph after a cycle (v, w, v) has been detected, i.e.:

$$A_G(w) := A_G(w) \setminus \{v\} \qquad A_G(v) := A_G(v) \setminus \{w\}. \tag{18}$$

A major improvement of *Johnson's algorithm* is motivated by the following considerations. If the set of vertices in a path of a cycle C is a superset of the set of vertices of a cycle C', then $C \subseteq C'$. Therefore, we are able to disregard cycles before we have complete knowledge of the set of vertices in a cycle. Since cycles are developed along a path, we included a procedure which compares the current path with the set of already detected cycles. As we will see at the end of this section, the improvement in running time is mainly due to this extension.

A complete description of the extended version of *Johnson's algorithm* is given in the Appendix.

The algorithm finds cycles by constructing paths where no vertex appears twice. The current path is stored on a stack. Calling CIRCUIT() appends a vertex to the stack, a return from CIRCUIT() causes the vertex on top of the stack to be removed. To prevent a vertex from being added more than once to the stack, a vertex is blocked when CIRCUIT() is called. Unblocking of vertices is done before a return from CIRCUIT(), by calling procedure UNBLOCK(). Procedure CIRCUIT() returns value TRUE if a cycle has been found on the current stack, and FALSE otherwise. The details of procedures CIRCUIT() and UNBLOCK() and the *main procedure* are discussed in [3].

Procedure SIMPLE_CYCLE() detects Single–Vertex and Two–Vertex cycles as discussed earlier in this section.

Each time a cycle is found, it is added to a list. Note that we are not interested in finding all cycles of the graph. Rather, we want to find only those cycles which are not subsets of other cycles (Lemma 1). Therefore, before a cycle is added to the list of relevant cycles, we call a procedure UPDATE_CYCLE() which checks if the new cycle is a superset of an already included cycle. In this case the new cycle can be disregarded. If the new cycle is a subset of some existing cycles, those cycles are eliminated and the new cycle is included into the list.

Procedure CHECK_STACK() tests the current stack against the existing list of cycles. If the contents of the stack is a superset of any already detected cycle, the search along the current stack

is stopped.

5.2 Evaluation of the Algorithm

The following examples will demonstrate how the performance of *Johnson*'s algorithm is improved with the presented extensions. We compare the discussed algorithm with a version where the procedures SIMPLE_CYCLE() and CHECK_STACK() are not included. This version corresponds to the original *Johnson*'s algorithm. We show examples of directed graphs where the adjacency structure is generated randomly with $Prob[(u,v) \in E] = p$ $(u,v \in V, 0 < p < 1)$. We do not consider Single–Vertex cycles, since Single–Vertex cycles result in a deletion of a vertex. Table 1 and Table 2 show the results for a graph with 10 and 20 vertices, respectively. The first column of the table provides the value of probability p, the second column the number of cycles in set $\tilde{C}$. Note that both algorithms provide the same output, i.e., the set $\tilde{C}$. The last two columns show the results for *Johnson*'s algorithm and the new version of the cycle finding algorithm. For each version we provide the number of cycles in the graph which are generated during the execution of the algorithm and the running times of each algorithm. We terminated program runs if their CPU time exceeded 10^5 seconds. The algorithm is implemented in C and run on a Sun Sparc workstation. We see in Tables 1 and 2 that the running time of *Johnson*'s algorithm increases fast and becomes soon computationally impractical. Note that Johnson's algorithm wastes running time by examining a very high number of cycles even when the size of set $\tilde{C}$ is small.

$\|V\| = 10$			Johnson	New
$p = 0.2$	$\|\tilde{C}\| = 1$	Cycles examined:	1	1
		CPU time (*sec*):	< 0.1	< 0.1
$p = 0.4$	$\|\tilde{C}\| = 12$	Cycles examined:	520	16
		CPU time (*sec*):	0.2	< 0.1
$p = 0.6$	$\|\tilde{C}\| = 21$	Cycles examined:	15,287	22
		CPU time (*sec*):	9.0	< 0.1
$p = 0.8$	$\|\tilde{C}\| = 27$	Cycles examined:	117,004	27
		CPU time (*sec*):	36.5	< 0.1

Table 2: Finding Cycles in a Graph with 10 Vertices.

$\|V\| = 20$			Johnson	New
$p = 0.2$	$\|\tilde{C}\| = 17$	Cycles examined:	3,718	39
		CPU time (*sec*):	3.4	0.2
$p = 0.4$	$\|\tilde{C}\| = 64$	Cycles examined:	—	70
		CPU time (*sec*):	$> 10^5$	0.2
$p = 0.6$	$\|\tilde{C}\| = 113$	Cycles examined:	—	127
		CPU time (*sec*):	$> 10^5$	0.4
$p = 0.8$	$\|\tilde{C}\| = 129$	Cycles examined:	—	130
		CPU time (*sec*):	$> 10^5$	0.1

Table 2: Finding Cycles in a Graph with 20 Vertices.

6 Special Network Topologies

The computational complexity of the buffer allocation algorithm presented in section 4 increases with the size of $\tilde{C}$, the set of cycles where no cycle is a subset of any other. For some networks the

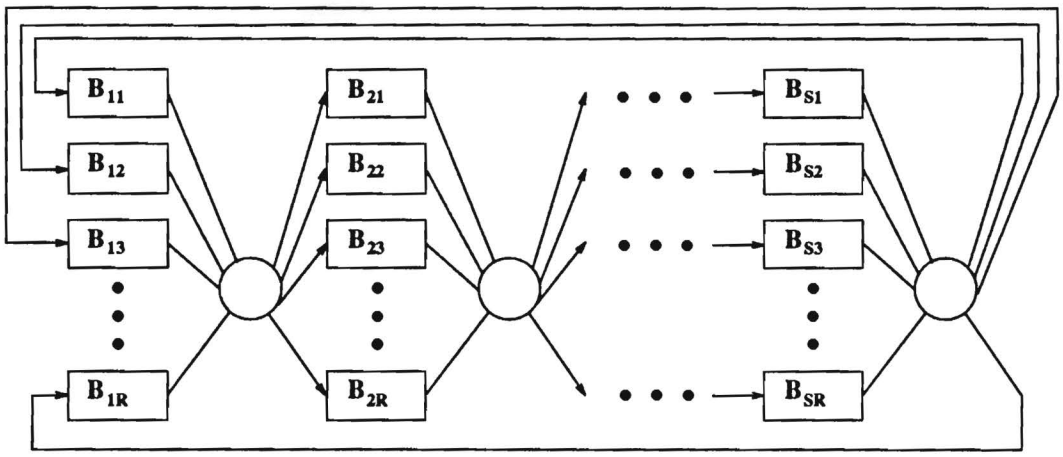


Figure 2: Tandem Network.

size of $\tilde{C}$ can make the optimization algorithm impractical. As a worst case, consider the network in Figure 2. Here $\tilde{C}$ is identical with the set of all buffer cycles in the network. Note that in the network in Figure 2 the set $\tilde{C}$ contains R^S cycles. In the following we show that the effort to obtain an optimal buffer allocation can be greatly reduced, if we take advantage of structural properties of the network.

Definition 5 *Given a multiple chain queueing network* Γ.

1. *A* tandem sequence *is a sequence of stations* $(\kappa_1, \kappa_2, \ldots, \kappa_{s-1}, \kappa_s)$ *such that*

$$\begin{aligned} &\mathcal{R}_{\kappa_v} = \mathcal{R}_{\kappa_w} && (1 \leq v, w \leq s) \\ &(\forall r \in \mathcal{R}_{\kappa_v})(p_{\kappa_v \kappa_{v+1}, r} = 1) && (1 \leq v < s) \\ &(\forall r \in \mathcal{R}_{\kappa_v})(p_{i \kappa_{v+1}, r} = 0) && (1 \leq v < s, i \neq \kappa_v). \end{aligned} \tag{19}$$

2. *A* tandem network Γ *with* S *stations is a tandem sequence* $(\kappa_1, \kappa_2, \ldots, \kappa_S, \kappa_{S+1})$ *where* $\kappa_1 = \kappa_{S+1}$.

In the following Lemma we show that an optimal buffer allocation for a tandem network assigns capacities to the buffers of only one station.

Lemma 3 *For a tandem network* Γ *with* S *stations and* R *routing chains, the following buffer allocations* $\hat{\Phi}_i$ $(i \in \mathcal{N})$ *are optimal:*

$$\hat{\Phi}_i(B_{jr}) = \begin{cases} K_r & \text{if } j = i \text{ and } K \geq S \text{ and } K - K_r \geq S \\ K - S & \text{if } j = i \text{ and } K \geq S > K - K_r \\ 0 & \text{if } j = i \text{ and } S > K \\ 0 & \text{if } j \neq i \end{cases} \tag{20}$$

$$\text{for } j \in \mathcal{N} \text{ and } r \in \mathcal{R}.$$

Proof:

1. $\hat{\Phi}_i$ *is a deadlock free allocation*

   Let us first consider the case: $S > K$. In this case, no state exists where all stations of Γ are occupied. Hence, for each cycle C there exists a chain $r' \in \mathcal{R}$ with:

$$|S_{r'}^{(C)}| > K_{r'}, \tag{21}$$

   which satisfies equation (6). For the other cases, consider $i = \kappa$ $(1 \leq \kappa \leq S)$. In a tandem network, each buffer cycle C has to contain a buffer from station κ, say $B_{\kappa\tau}$ $(\tau \in \mathcal{R})$. Since:

$$\begin{aligned} &\hat{\Phi}_\kappa(B_{\kappa\tau}) = K_\tau && \wedge\ |S_\tau^{(C)}| > 0, && \text{if } K - K_\tau \geq S, \\ &\hat{\Phi}_\kappa(B_{\kappa\tau}) = K_\tau - (K - K_\tau) && \wedge\ |S_\tau^{(C)}| > S - (K - K_\tau), && \text{if } K - K_\tau < S. \end{aligned} \tag{22}$$

We obtain in either case:

$$\hat{\Phi}_\kappa(B_{\kappa\tau}) + |S_\tau^{(C)}| > K_\tau. \tag{23}$$

This satisfies the DLF condition for chain τ in Cycle C.

2. *$\hat{\Phi}_i$ is minimal*

If $S > K$ no capacity is allocated to any buffer, i.e., $\hat{\Phi}_i$ is minimal. For $K \geq S$ we will prove the minimality of allocations $\hat{\Phi}_i$ for chain r ($r \in \mathcal{R}$) by showing that any allocation which allocates less total capacity to the buffers than $\hat{\Phi}_i$ will result in a deadlock. First we consider all chains τ with $K_\tau > S$ and allocations Φ^{**} with $\sum_{i\in\mathcal{N}_\tau} \Phi^{**}(B_{i\tau}) < K_\tau$. Consider also $K_\tau = S$ and a buffer allocation Φ^* with $\sum_{i\in\mathcal{N}_\tau} \Phi^*(B_{i\tau}) < S$. Since:

$$\sum_{i\in S_\tau^{(C)}} \Phi^{**}(B_{i\tau}) \leq \sum_{i\in S_\tau^{(C)}} \Phi^*(B_{i\tau}) + (K_\tau - S) \tag{24}$$

Therefore, a violation of the DLF condition by chain τ for $K_\tau = S$ and Φ^* implies a violation of the DLF condition by chain τ in the same buffer cycle for $K_\tau > S$ and Φ^{**}. Thus, we only need to consider $K_\tau \leq S$.

We distinguish five cases:

(a) *$K_\tau = S$ and $K - K_\tau = S$*

We will show that any allocation Φ^* with $\sum_{i\in\mathcal{N}_\tau} \Phi^*(B_{i\tau}) = K_\tau - 1$ contains a deadlock. Since a deadlock in a network with more than two routing chains can always be reduced to a deadlock in a network with two routing chains we can assume without loss of generality that $R = 2$ with two chains r and r'.

Assume a buffer cycle C such that $|S_r^{(C)}| > 0$ and $|S_{r'}^{(C)}| > 0$. Note that $S_r^{(C)} \cap S_{r'}^{(C)} = \emptyset$, and $S_r^{(C)} \cup S_{r'}^{(C)} = \mathcal{N}$.

We can derive a buffer cycle C' from C with $S_r^{(C')} = S_{r'}^{(C)}$ and $S_{r'}^{(C')} = S_r^{(C)}$.

The DLF condition of cycle C for chain r reads:

$$\sum_{i\in S_r^{(C)}} \Phi^*(B_{ir}) + |S_r^{(C)}| > K_r \tag{25}$$

Since $S_r^{(C')} = \mathcal{N} \setminus S_r^{(C)}$ the DLF condition of cycle C' for chain r yields:

$$\sum_{i\in\mathcal{N}\setminus S_r^{(C)}} \Phi^*(B_{ir}) + (S - |S_r^{(C)}|) > K_r \tag{26}$$

Since $\sum_{i \in \mathcal{N}} \Phi^*(B_{ir}) = K_r - 1$ and $S = K_r$ we can rewrite equation (26) as:

$$(K_r - 1 - \sum_{i \in S_r^{(C)}} \Phi^*(B_{ir})) + (K_r - |S_r^{(C)}|) > K_r \tag{27}$$

Equation (27) is equivalent to:

$$\sum_{i \in S_r^{(C)}} \Phi^*(B_{ir}) + |S_r^{(C)}| < K_r - 1 \tag{28}$$

From equations (25) and (28) we see that the DLF conditions for chain r cannot be satisfied in both cycles C and C'. The same property can be derived for routing chain r'.

We now show that there exist two cycles $\tilde{C}$ and $\overline{C}$ such that the deadlock conditions for routing chain r are violated in both cycles. Let us partition $\mathcal{N}$, the set of all stations, into two subsets $\mathcal{Z}$ and $\mathcal{NZ}$, the set of stations with zero capacity buffers in chain r, and the set of stations with nonzero capacity buffers in chain r, respectively. $\mathcal{Z}$ and $\mathcal{NZ}$ are given by:

$$\begin{aligned} \mathcal{Z} &= \{i \mid \Phi^*(B_{ir}) = 0\} \\ \mathcal{NZ} &= \{i \mid \Phi^*(B_{ir}) > 0\} \end{aligned}$$

Now we partition $\mathcal{NZ}$ further into two disjoint sets $\mathcal{X}$ and $\mathcal{Y}$ with the following algorithm:

begin
  $\mathcal{X} := \emptyset$;
  $\mathcal{Y} := \emptyset$;
  while ($\mathcal{NZ} \neq \emptyset$) **do**
    begin
      Select v such that $\Phi^*(B_{vr}) = \max_{j \in \mathcal{NZ}}\{\Phi^*(B_{jr}\}$;
      if ($\sum_{i \in \mathcal{X}} \Phi^*(B_{ir}) + |\mathcal{X}| + \Phi^*(B_{vr}) + 1 > S$) **then**
        $\mathcal{X} := \mathcal{X} \cup \{v\}$;
      else
        $\mathcal{Y} := \mathcal{Y} \cup \{v\}$;
      endif
      $\mathcal{NZ} := \mathcal{NZ} \setminus \{v\}$;
  endwhile
end

Per construction of $\mathcal{X}$ the buffers from stations $\mathcal{X}$ yield:

$$\sum_{i \in \mathcal{X}} \Phi^*(B_{ir}) + |\mathcal{X}| < S \tag{29}$$

Now let us assume:

$$\sum_{i \in \mathcal{Y}} \Phi^*(B_{ir}) + |\mathcal{Y}| > S \tag{30}$$

Take an arbitrary station $k \in \mathcal{Y}$. Because of the construction of $\mathcal{X}$ and $\mathcal{Y}$ we obtain:

$$\sum_{i \in \mathcal{X}} \Phi^*(B_{ir}) + |\mathcal{X}| + \Phi^*(B_{kr}) + 1 > S \tag{31}$$

Note that by construction sets $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$ have the following properties:

$$S = |\mathcal{X}| + |\mathcal{Y}| + |\mathcal{Z}| \tag{32}$$

$$S - 1 = \sum_{i \in \mathcal{X}} \Phi^*(B_{ir}) + \sum_{i \in \mathcal{Y}} \Phi^*(B_{ir}) \tag{33}$$

Adding equations (30) and (31) we obtain after simplifying the sum with equations (32) and (33):

$$\Phi^*(B_{kr}) > \mathcal{Z} \tag{34}$$

However, this is a contradiction to the properties in equations (32) and (33). Therefore, set $\mathcal{Y}$ must satisfy:

$$\sum_{i \in \mathcal{Y}} \Phi^*(B_{ir}) + |\mathcal{Y}| \leq S \tag{35}$$

Note that summing equations (32) and (33) yields:

$$2S - 1 = |\mathcal{X}| + |\mathcal{Y}| + |\mathcal{Z}| + \sum_{i \in \mathcal{X}} \Phi^*(B_{ir}) + \sum_{i \in \mathcal{Y}} \Phi^*(B_{ir}) \tag{36}$$

Thus, we can partition $\mathcal{Z}$ into two disjoint sets $\mathcal{Z}_1$ and $\mathcal{Z}_2$ such that:

$$\sum_{i \in (\mathcal{X} \cup \mathcal{Z}_1)} \Phi^*(B_{ir}) + |\mathcal{X}| + |\mathcal{Z}_1| \leq S \tag{37}$$

$$\sum_{i \in (\mathcal{Y} \cup \mathcal{Z}_2)} \Phi^*(B_{ir}) + |\mathcal{Y}| + |\mathcal{Z}_2| \leq S \tag{38}$$

Now we define a cycle $\overline{C}$ with $S_r^{(\overline{C})} = (\mathcal{X} \cup \mathcal{Z}_1)$ and $S_{r'}^{(\overline{C})} = (\mathcal{Y} \cup \mathcal{Z}_2)$, and a cycle $\tilde{C}$ with $S_r^{(\tilde{C})} = (\mathcal{Y} \cup \mathcal{Z}_2)$ and $S_{r'}^{(\tilde{C})} = (\mathcal{X} \cup \mathcal{Z}_1)$. Because of equations (37) and (38) routing chain r does not satisfy the DLF condition in either cycle $\overline{C}$ or $\tilde{C}$. Because of equations (25) and (28), routing chain r' cannot satisfy the DLF conditions for both cycles $\overline{C}$ and $\tilde{C}$. Therefore, either cycle $\overline{C}$ or $\tilde{C}$ contains a deadlock.

(b) $K_r = S$ *and* $K - K_r < S$

Since in this case

$$|S_r^{(C)}| > S - (K - K_r) \tag{39}$$

we can construct a tandem network $\overline{\Gamma}$ from Γ with $\overline{S}$ stations and $\overline{K}_\rho$ jobs in chain ρ with:

$$\begin{aligned} \overline{S} &= K - K_r \\ \overline{\mathcal{R}} &= \mathcal{R} \\ \overline{K}_r &= K_r - (S - (K - K_r)) = K - S \\ \overline{K}_\rho &= K_\rho \qquad (\rho \in \mathcal{R} \setminus \{r\}) \end{aligned} \tag{40}$$

Per construction, Γ contains a deadlock if $\overline{\Gamma}$ contains a deadlock. Since $(K - K_r) \geq \overline{S}$, case (a) applies.

(c) $K_r < S$ *and* $K - K_r = K_r$

Since in this case

$$\sum_{\rho \in \mathcal{R} \setminus \{r\}} |S_\rho^{(C)}| > S - K_r \tag{41}$$

we can construct a tandem network $\overline{\Gamma}$ from Γ with $\overline{S}$ stations and $\overline{K}_\rho$ jobs in chain ρ with:

$$\begin{aligned} \overline{S} &= K_r \\ \overline{\mathcal{R}} &= \mathcal{R} \\ \overline{K}_r &= K_r \\ \overline{K}_\rho &= K_\rho - k_\rho \qquad (\rho \in \mathcal{R} \setminus \{r\}) \end{aligned} \tag{42}$$

with:

$$\sum_{\rho \in \mathcal{R} \setminus \{r\}} k_\rho = S - K_r \tag{43}$$

Per construction, Γ contains a deadlock if $\overline{\Gamma}$ contains a deadlock. Since $K_r = \overline{S}$, case (a) applies.

(d) $K_r < S$ *and* $K - K_r < K_r$

We first perform the same construction as in case (b) and obtain a network $\overline{\Gamma}$ with $\overline{S} = K - K_r$. Then we perform the construction as in case (c) and obtain obtain a network $\tilde{\Gamma}$ $\tilde{S} = K_r$.

□

If the network has only a single routing chain $K = K_1$, where K_1 denotes the total number of jobs in the chain. Therefore, an optimal buffer allocation for a single chain network can be obtained from Lemma 3 by:

$$\hat{\Phi}_i(B_j) = \begin{cases} K - S & \text{if } j = i \\ 0 & \text{if } j \neq i \end{cases} \tag{44}$$

From the proof of Lemma 3, we can follow immediately:

Corollary 2 *A buffer allocation Φ^* for a tandem network with S stations and R routing chains with $(K - K_r) \geq S$ (for $1 \leq r \leq R$) is optimal, if and only if there exists an $i \in \mathcal{N}$ such that for all $j \in \mathcal{N}$ and $r \in \mathcal{R}_j$:*

$$\Phi^*(B_{jr}) = \hat{\Phi}_i(B_{jr}) \tag{45}$$

With Lemma 3 we are able to give an optimal buffer allocation without applying the algorithm of section 4. However, the Lemma is only applicable for the special class of tandem networks. In the following we show how to take advantage of the regularity of networks which contain a tandem sequence, i.e, contain a network like the network in Figure 2 as a subnetwork. The following Lemma prepares the so-called *Reduction Lemma* which will be presented at the end of this section. The *Reduction Lemma* allows to perform the allocation algorithm from section 4 on a network where the tandem sequence is substituted by a single station. The results from the reduced network can be used to obtain an optimal allocation of the original network.

Lemma 4 *Given an optimal buffer allocation Φ^* for a network Γ. If Γ contains a tandem sequence of S stations $(\kappa_1, \kappa_2, \ldots, \kappa_{S-1}, \kappa_S)$, the following buffer allocation $\hat{\Phi}$ is optimal:*

$$\hat{\Phi}(B_{ir}) = \begin{cases} \sum_{v=1}^{S} \Phi^*(B_{\kappa_v r}) & \text{if } i = \kappa_1 \\ 0 & \text{if } i = \kappa_v \text{ and } 1 < v \leq S \\ \Phi^*(B_{ir}) & \text{otherwise} \end{cases} \tag{46}$$

$$\text{for } i \in \mathcal{N}, r \in \mathcal{R}_i.$$

Proof: $\hat{\Phi}$ is per definition minimal. It remains to be shown that $\hat{\Phi}$ yields a deadlock free network. Assume the network with allocation $\hat{\Phi}$ is in a deadlock. The deadlock has to be along a cycle which includes one buffer from each of stations $(\kappa_1, \kappa_2, \ldots, \kappa_{S-1}, \kappa_S)$. Otherwise, Γ has a deadlock under Φ^*. Note that Φ^* is equal to $\hat{\Phi}$ for buffers of stations not in the tandem sequence. Let the buffer

cycle which contains a possible deadlock be given by:

$$\widehat{C} = (B_{\sigma_1\rho_1}, B_{\sigma_2\rho_2}, \ldots, B_{\kappa_1\tau_1}, B_{\kappa_2\tau_2}, \ldots, B_{\kappa_{S-1}\tau_{S-1}}, B_{\kappa_S\tau_S}, \ldots, B_{\sigma_{t-1}\rho_{t-1}}, B_{\sigma_t\rho_t}) \tag{47}$$

Consider the tandem network Ψ obtained from Γ by eliminating the stations not belonging to the tandem sequence. Let $\mathcal{N}_\Psi$, $\mathcal{R}_\Psi$, $K_{\Psi,r}$ denote the set of stations, the set of classes and the number of jobs, respectively, in Ψ, with:

$$\begin{aligned} \mathcal{N}_\Psi &= \{B_{\kappa_1\tau_1}, B_{\kappa_2\tau_2}, \ldots, B_{\kappa_{S-1}\tau_{S-1}}, \ldots, B_{\sigma_S\rho_S}\}, \\ \mathcal{R}_\Psi &= \mathcal{R}_{\kappa_1}, \\ K_{\Psi,r} &= K_r - (\sum_{(1<u\leq t)\wedge(\sigma_u \in \overline{S}_r^{(\widehat{C})})} \Phi^*(B_{\sigma_u r}) + |\overline{S}_r^{(\widehat{C})}|), \end{aligned} \tag{48}$$

where:

$$\overline{S}_r^{(C)} = S_r^{(C)} \setminus \{\kappa_1, \kappa_2, \ldots, \kappa_{S-1}, \kappa_S\}. \tag{49}$$

Any allocation Φ for network Γ defines an allocation for Ψ. Per definition of Ψ, allocation $\hat{\Phi}$ for Ψ violates the DLF conditions for cycle $\tilde{C}$ with:

$$\tilde{C} = (B_{\kappa_1\tau_1}, B_{\kappa_2\tau_2}, \ldots, B_{\kappa_{S-1}\tau_{S-1}}, B_{\sigma_S\rho_S}), \tag{50}$$

if $\widehat{C}$ in Γ under $\hat{\Phi}$ violates the DLF conditions. According to Lemma 3, we can find a minimal and deadlock free buffer allocation for Ψ with:

$$\hat{\Phi}(B_{ir}) = 0, \qquad \text{if } i = \kappa_v \text{ and } 1 < v \leq S. \tag{51}$$

Therefore, if $\hat{\Phi}$ contains a deadlock, then the assumption that Φ^* is optimal for Γ does not hold. □

Lemma 5 (Reduction Lemma) *Given a network Γ with a tandem sequence $(\kappa_1, \kappa_2, \ldots, \kappa_{S-1}, \kappa_S)$ with $r \in \mathcal{R}_{\kappa_1}$. Obtain a network Γ' by substituting the tandem sequence in Γ by a single station $\tilde{\kappa}$. Let $\tilde{\Phi}$ be a minimal allocation for Γ'. Define for all $r \in \mathcal{R}_{\tilde{\kappa}}$*

$$M_{\tilde{\kappa}r} = \max\{ \sum_{\rho \in \mathcal{R}\setminus\{r\}} (K_\rho - \sum_{(1<u\leq t)\wedge(\sigma_u \in S_r^{(C)})} \tilde{\Phi}(B_{\sigma_u\rho}) - |S_\rho^{(C)}|) \mid C \text{ Cycle in } \Gamma',\ B_{\tilde{\kappa}r} \in C\}. \tag{52}$$

If the condition holds that:

$$M_{\tilde{\kappa}r} + \tilde{\Phi}(B_{\tilde{\kappa}r}) \geq S \qquad \text{for all } r \in \mathcal{R}_{\tilde{\kappa}}, \tag{53}$$

then the following buffer allocation Φ^* is optimal for Γ:

$$\Phi^*(B_{ir}) = \begin{cases} \tilde{\Phi}(B_{\tilde{\kappa}r}) & \text{if } i = \kappa_1 \text{ and } M_{\tilde{\kappa}r} \geq S-1 \\ \tilde{\Phi}(B_{\tilde{\kappa}r}) - (S-1) + M_{\tilde{\kappa}r} & \text{if } i = \kappa_1 \text{ and } M_{\tilde{\kappa}r} < S-1 \\ 0 & \text{if } i = \kappa_v \text{ and } 1 < v \leq S \\ \tilde{\Phi}(B_{ir}) & \text{otherwise} \end{cases} \tag{54}$$

$$\text{for } i \in \mathcal{N}, r \in \mathcal{R}_i.$$

Proof:

1. Φ^* *is a deadlock free allocation*

   We only have to consider cycles which include buffers from stations of the tandem sequence. Assume a cycle C' in Γ' which includes buffer $B_{\tilde{\kappa}\tau_1}$ ($\tau_1 \in \mathcal{R}_{\tilde{\kappa}}$):

   $$C' = (B_{\sigma_1\rho_1}, B_{\sigma_2\rho_2}, \ldots, B_{\tilde{\kappa}\tau_1}, \ldots, B_{\sigma_{t-1}\rho_{t-1}}, B_{\sigma_t\rho_t}) \tag{55}$$

   C' corresponds to a set of cycles in Γ:

   $$\mathcal{C}'' = \{(B_{\sigma_1\rho_1}, B_{\sigma_2\rho_2}, \ldots, B_{\kappa_1\tau_1}, B_{\kappa_2\tau_2}, \ldots, B_{\kappa_S\tau_S}, \ldots, B_{\sigma_{t-1}\rho_{t-1}}, B_{\sigma_t\rho_t}) \mid \tau_v \in \mathcal{R}_{\tau_1}, 1 \leq v \leq S\} \tag{56}$$

   (a) $M_{\tilde{\kappa}r} \geq S-1$

   Then for all $C'' \in \mathcal{C}''$ and for all $r \in \mathcal{R}$ the following holds:

   $$\sum_{(1<u\leq t)\wedge(\sigma_u \in S_r^{(C')})} \Phi^*(B_{\sigma_u r}) = \sum_{(1<u\leq t)\wedge(\sigma_u \in S_r^{(C'')})} \tilde{\Phi}(B_{\sigma_u r}), \tag{57}$$

   and:

   $$|S_r^{(C')}| \leq |S_r^{(C'')}|, \tag{58}$$

   Since C' is deadlock free in Γ' with $\tilde{\Phi}$, it follows from (57) and (58) that all $C'' \in \mathcal{C}''$ are deadlock free in Γ with Φ^*.

   (b) $M_{\tilde{\kappa}r} < S-1$

   Recall from (54) that in this case $\Phi^*(B_{\tilde{\kappa}r}) < \tilde{\Phi}(B_{\tilde{\kappa}r})$. If the DLF condition for cycle C' in Γ' is satisfied by chain $\rho \neq \tau_1$, then equations (57) and (58) hold for chain ρ. But

then, the DLF condition for $C'' \in \mathcal{C}''$ in Γ is satisfied by chain $\rho \neq \tau_1$. Now assume that chain τ_1 satisfies the DLF condition for cycle C' in Γ', i.e.,

$$\sum_{(1<u\leq t)\wedge(\sigma_u \in S_{\tau_1}^{(C')})} \Phi^*(B_{\sigma_u \tau_1}) + |S_{\tau_1}^{(C')}| > K_{\tau_1} \tag{59}$$

Since $M_{\tilde{\kappa}r} < S - 1$, at least $((S-1) - M_{\tilde{\kappa}r})$ of the servers of stations $(\kappa_2, \kappa_3, \ldots, \kappa_S)$ have to be occupied by jobs of chain τ_1 if a deadlock persists in a cycle $C'' \in \mathcal{C}''$. But then:

$$|S_{\tau_1}^{(C'')}| \geq |S_{\tau_1}^{(C')}| + ((S-1) - M_{\tilde{\kappa}\tau_1}) \tag{60}$$

Note that from (54):

$$\sum_{(1<u\leq t)\wedge(\sigma_u \in S_{\tau_1}^{(C')})} \Phi^*(B_{\sigma_u \tau_1}) = \sum_{(1<u\leq t)\wedge(\sigma_u \in S_{\tau_1}^{(C'')})} \tilde{\Phi}(B_{\sigma_u \tau_1}) - ((S-1) - M_{\tilde{\kappa}\tau_1}). \tag{61}$$

From equations (60) and (61) we see immediately, that chain τ_1 satisfies the DLF condition for all $C'' \in \mathcal{C}''$ of Γ.

2. Φ^* *is minimal*

Because of Lemma 4 we only need to consider allocations which assign non-zero capacity only to station κ_1 and zero capacity to stations κ_v $(1 < v \leq S)$ in Γ. Since $\tilde{\Phi}$ is optimal for Γ', network Γ' will have a deadlock if we assign less capacity than $\tilde{\Phi}(B_{\tilde{\kappa}\tau_1})$ to buffer $B_{\tilde{\kappa}\tau_1}$. Take a cycle $\widehat{C}$ in Γ' with, which satisfies the DLF condition for chain $\tilde{\kappa}$ if a capacity of $\tilde{\Phi}(B_{\tilde{\kappa}\tau_1})$ is assigned to $B_{\tilde{\kappa}\tau_1}$, but does not so if the capacity is $(\tilde{\Phi}(B_{\tilde{\kappa}\tau_1}) - 1)$. $\widehat{C}$ has the same structure as the cycle in (55). Let $\widehat{\widehat{\mathcal{C}}}$ be the set of cycles in Γ which correspond to $\widehat{C}$ in Γ'. $\widehat{\widehat{\mathcal{C}}}$ has the same structure as set $\mathcal{C}'$ in (56). For a cycle $\widehat{\widehat{C}} \in \widehat{\widehat{\mathcal{C}}}$ we distinguish two cases:

(a) $M_{\tilde{\kappa}r} \geq S - 1$

In this case, the servers of stations $(\kappa_2, \kappa_3, \ldots, \kappa_S)$ can be occupied by jobs of chains from $\mathcal{R} \setminus \{r\}$ such that no $\rho \in \mathcal{R} \setminus \{r\}$ satisfies the DLF condition for cycle $\widehat{\widehat{C}}$. Then:

$$\sum_{(1<u\leq t)\wedge(\sigma_u \in S_{\tau_1}^{(\widehat{\widehat{C}})})} \Phi^*(B_{\sigma_u \tau_1}) = \sum_{(1<u\leq t)\wedge(\sigma_u \in S_{\tau_1}^{(\widehat{C})})} \tilde{\Phi}(B_{\sigma_u \tau_1}), \tag{62}$$

and:

$$|S_{\tau_1}^{(\widehat{\widehat{C}})}| = |S_{\tau_1}^{(\widehat{C})}|, \tag{63}$$

Therefore, any allocation with:

$$\Phi^*(B_{\kappa_1\tau_1}) < \tilde{\Phi}(B_{\tilde{\kappa}\tau_1}) \tag{64}$$

does not satisfy the DLF condition for chain τ_1, and thus, a deadlock persists for all $\hat{\hat{C}} \in \hat{\hat{\mathcal{C}}}$.

(b) $M_{\tilde{\kappa}r} < S - 1$

Here $M_{\tilde{\kappa}\tau_1}$ of the servers of stations $(\kappa_2, \kappa_3, \ldots, \kappa_S)$ can be occupied by jobs of chains from $\mathcal{R}\backslash\{r\}$ without satisfying the DLF condition for any chain $\rho \in \mathcal{R}\backslash\{r\}$. If condition (53) holds, then the remaining $((S-1) - M_{\tilde{\kappa}\tau_1})$ servers can be occupied by jobs from chain τ_1. Then:

$$\begin{aligned}
&\sum_{(1<u\leq t)\wedge(\sigma_u \in S_{\tau_1}^{(\hat{\hat{C}})}\backslash\{\kappa_1\})} \Phi^*(B_{\sigma_u\tau_1}) + \Phi^*(B_{\kappa_1\tau_1}) + |S_{\tau_1}^{(\hat{\hat{C}})}| \\
&= \sum_{(1<u\leq t)\wedge(\sigma_u \in S_{\tau_1}^{(\hat{C})})\backslash\{\kappa_1\}} \Phi^*(B_{\sigma_u\tau_1}) + (\tilde{\Phi}(B_{\tilde{\kappa}\tau_1}) - (S-1) + M_{\tilde{\kappa}\tau_1}) + (|S_{\tau_1}^{(\hat{C})}| + (S-1) - M_{\tilde{\kappa}\tau_1}) \\
&= \sum_{(1<u\leq t)\wedge(\sigma_u \in S_{\tau_1}^{(\hat{C})}\backslash\{\tilde{\kappa}\})} \tilde{\Phi}(B_{\sigma_u\tau_1}) + \tilde{\Phi}(B_{\tilde{\kappa}\tau_1}) + |S_{\tau_1}^{(\hat{C})}|
\end{aligned} \tag{65}$$

But then, any allocation with:

$$\Phi^*(B_{\kappa_1\tau_1}) < \tilde{\Phi}(B_{\tilde{\kappa}\tau_1}) - (S-1) + M_{\tilde{\kappa}\tau_1} \tag{66}$$

does not satisfy the DLF condition for chain τ_1. We follow from the construction of $\hat{\hat{C}}$ that network Γ contains a deadlock for all $\hat{\hat{C}} \in \hat{\hat{\mathcal{C}}}$. □

7 Examples

In this section we show applications of the deadlock free buffer allocation algorithm from section 4. In the first two examples we demonstrate the steps of the allocation algorithm for a single chain queueing network (Example 1), and for a multiple chain queueing network (Example 2). The third and fourth example show applications of the *Reduction Lemma* (Lemma 5). All linear programs shown in the examples were solved with the programming package LINDO [4].

7.1 Example 1

Note that the deadlock free buffer allocation algorithm for single chain queueing networks in [2] is only applicable to so-called cacti networks, i.e., queueing networks where no two cycles have more than one buffer in common. Our algorithm does not have any restrictions on the structure of the network. We will show the steps of the buffer allocation algorithm for the network shown in Figure 3. Since we have a single chain network, the chain identifying index has been omitted, i.e., B_i

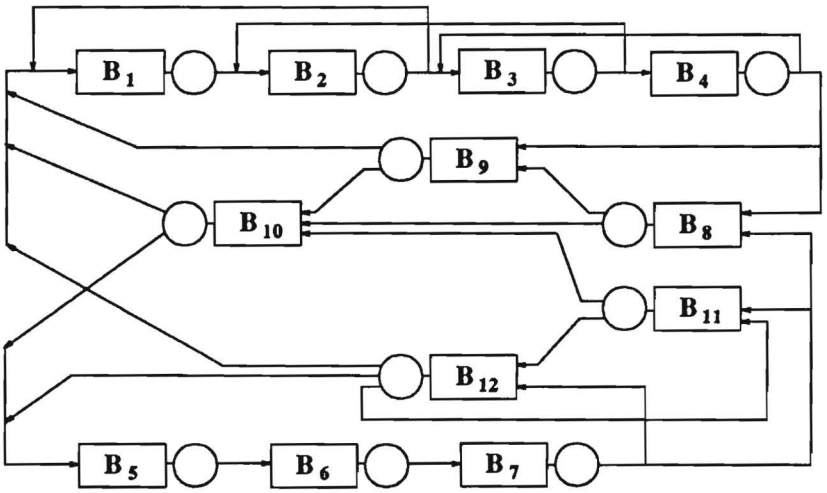


Figure 3: Single Chain Queueing Network.

denotes the single buffer of station i. Although the networks contains 28 elementary cycles, the cycle finding algorithm given in section 5 produces only 7 cycles which are relevant for a deadlock free buffer allocation. The cycles are:

$$
\begin{aligned}
C_1 &= (B_1, B_2, B_1) \\
C_2 &= (B_2, B_3, B_2) \\
C_3 &= (B_3, B_4, B_3) \\
C_4 &= (B_5, B_6, B_7, B_8, B_{10}, B_5) \\
C_5 &= (B_5, B_6, B_7, B_{10}, B_{11}, B_5) \\
C_6 &= (B_5, B_6, B_7, B_{12}, B_5) \\
C_7 &= (B_{11}, B_{12}, B_{11})
\end{aligned}
$$

The integer program corresponding to the set of cycles has the following structure. Recall from section 4.2 that we use b_i to denote the variable for the capacity assigned to buffer B_i.

minimize $b_1 + b_2 + b_3 + b_4 + b_5 + b_6 + b_7 + b_8 + b_9 + b_{10} + b_{11} + b_{12}$

subject to

$$\begin{aligned}
b_1 + b_2 &\geq K - 1 \\
b_2 + b_3 &\geq K - 1 \\
b_3 + b_4 &\geq K - 1 \\
b_5 + b_6 + b_7 + b_8 + b_{10} &\geq K - 4 \\
b_5 + b_6 + b_7 + b_{10} + b_{11} &\geq K - 4 \\
b_5 + b_6 + b_7 + b_{12} &\geq K - 3 \\
b_{11} + b_{12} &\geq K - 1 \\
b_i &\geq 0 \qquad (i = 1, 2, \ldots, 12).
\end{aligned}$$

Note that we do not need (0,1)-variables y_{lr} for single chain networks (equation (16)). Table 3 presents the solution for different values of the total number of jobs K. B^* denotes the sum of all buffer allocations. We show both the optimal solution obtained from solving the integer program, and the results obtained from the heuristic method, discussed in subsections 4.2.1 and 4.2.2, respectively. Note the accuracy of the results from heuristic method in Table 3. We obtain an optimal deadlock free buffer allocation Φ^* for the network in Figure 3 with:

$$\Phi^*(B_i) = b_i \quad (i = 1, 2, \ldots, 12)$$

For $K = 4$, the condition of Corollary 1 (equation (10) is satisfied for the following cycles:

$$\begin{aligned}
C_4 &= (B_5, B_6, B_7, B_8, B_{10}, B_5) \\
C_5 &= (B_5, B_6, B_7, B_{10}, B_{11}, B_5) \\
C_6 &= (B_5, B_6, B_7, B_{12}, B_5)
\end{aligned}$$

Therefore, we can eliminate the constraints in the integer program which were derived from these cycles.

	$K = 4$		$K = 10$		$K = 35$	
Φ^*	optimal	heuristic	optimal	heuristic	optimal	heuristic
B^*	9	9	33	34	133	134
b_1	0	0	0	0	0	0
b_2	3	3	9	9	34	34
b_3	0	3	0	9	0	34
b_4	3	0	9	0	34	0
b_5	0	0	2	7	0	32
b_6	0	0	0	0	0	0
b_7	0	0	0	0	0	0
b_8	0	0	4	0	0	0
b_9	0	0	0	0	0	0
b_{10}	0	0	0	0	31	0
b_{11}	0	0	4	9	0	34
b_{12}	3	3	5	0	34	0

Table 3: Results for Example 1.

7.2 Example 2

Figure 4 depicts a queueing network with three routing chains. In the first step of the algorithm, we construct a directed graph according to Lemma 2. The directed graph has 429 elementary cycles. However, the algorithm described in section 5 yields only 19 cycles which are relevant for the optimization:

$$
\begin{aligned}
C_1 &= (B_{1,1}, B_{2,1}, B_{1,1}) \\
C_2 &= (B_{9,2}, B_{10,2}, B_{9,2}) \\
C_3 &= (B_{5,3}, B_{8,3}, B_{10,3}, B_{6,3}, B_{5,3}) \\
C_4 &= (B_{2,1}, B_{4,1}, B_{10,1}, B_{2,1}) \\
C_5 &= (B_{2,1}, B_{4,1}, B_{5,1}, B_{10,2}, B_{2,1}) \\
C_6 &= (B_{2,1}, B_{4,1}, B_{5,2}, B_{8,3}, B_{10,3}, B_{2,1}) \\
C_7 &= (B_{2,1}, B_{4,1}, B_{5,2}, B_{10,2}, B_{2,1}) \\
C_8 &= (B_{2,1}, B_{4,1}, B_{7,2}, B_{10,2}, B_{2,1}) \\
C_9 &= (B_{4,2}, B_{5,1}, B_{8,3}, B_{10,3}, B_{3,3}, B_{4,2}) \\
C_{10} &= (B_{4,2}, B_{5,1}, B_{8,3}, B_{10,3}, B_{9,2}, B_{3,2}, B_{4,2}) \\
C_{11} &= (B_{4,2}, B_{5,1}, B_{10,2}, B_{3,3}, B_{4,2}) \\
C_{12} &= (B_{4,2}, B_{5,2}, B_{8,3}, B_{10,3}, B_{3,3}, B_{4,2}) \\
C_{13} &= (B_{4,2}, B_{5,2}, B_{8,3}, B_{10,3}, B_{9,2}, B_{3,2}, B_{4,2}) \\
C_{14} &= (B_{4,2}, B_{5,2}, B_{10,2}, B_{3,3}, B_{4,2}) \\
C_{15} &= (B_{4,2}, B_{10,1}, B_{3,3}, B_{4,2}) \\
C_{16} &= (B_{4,2}, B_{10,1}, B_{9,2}, B_{3,2}, B_{4,2})
\end{aligned}
$$

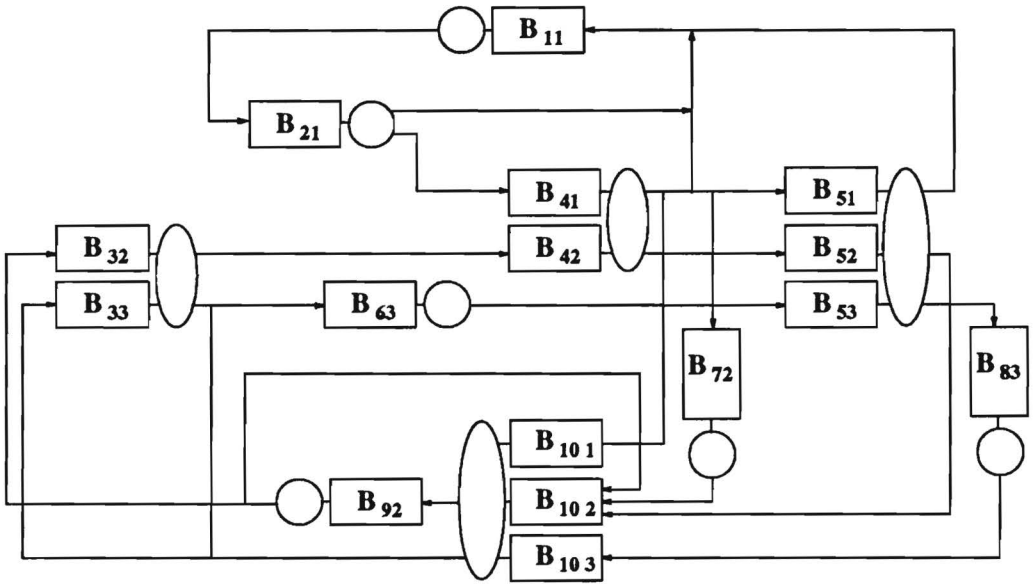


Figure 4: Multiple Chain Queueing Network.

$$C_{17} = (B_{4,2}, B_{7,2}, B_{10,2}, B_{3,3}, B_{4,2})$$
$$C_{19} = (B_{5,3}, B_{10,2}, B_{6,3}, B_{5,3})$$

To write the integer program we first have to establish the sets $S_r^{(C_l)}(r = 1, 2, 3;\ l = 1, 2, \ldots, 19)$. Then we obtain:

minimize $\mathrm{b}_{1,1} + \mathrm{b}_{2,1} + \mathrm{b}_{4,1} + \mathrm{b}_{5,1} + \mathrm{b}_{10,1} + \mathrm{b}_{3,2} + \mathrm{b}_{4,2} + \mathrm{b}_{5,2} + \mathrm{b}_{7,2} + \mathrm{b}_{9,2} + \mathrm{b}_{10,2} + \mathrm{b}_{3,3} + \mathrm{b}_{5,3} + \mathrm{b}_{6,3} + \mathrm{b}_{8,3} + \mathrm{b}_{10,3}$

subject to

$$\begin{array}{lll}
C_1: & \mathrm{b}_{1,1} + \mathrm{b}_{2,1} \geq K_1 - 1 \\
C_2: & \mathrm{b}_{9,2} + \mathrm{b}_{10,2} \geq K_2 - 1 \\
C_3: & \mathrm{b}_{5,3} + \mathrm{b}_{6,3} + \mathrm{b}_{8,3} + \mathrm{b}_{10,3} \geq K_3 - 3 \\
C_4: & \mathrm{b}_{2,1} + \mathrm{b}_{4,1} + \mathrm{b}_{10,1} \geq K_1 - 2 \\
C_5: & \mathrm{b}_{2,1} + \mathrm{b}_{4,1} + \mathrm{b}_{5,1} - (K_1 - 2) \cdot \mathrm{y}_{5,1} \geq 0 \\
 & \mathrm{b}_{10,2} - K_2 \cdot \mathrm{y}_{5,2} \geq 0 \\
 & \mathrm{y}_{5,1} + \mathrm{y}_{5,2} \geq 1 \\
C_6: & \mathrm{b}_{2,1} + \mathrm{b}_{4,1} - (K_1 - 1) \cdot \mathrm{y}_{6,1} \geq 0 \\
 & \mathrm{b}_{5,2} - K_2 \cdot \mathrm{y}_{6,2} \geq 0 \\
 & \mathrm{b}_{8,3} + \mathrm{b}_{10,3} - (K_3 - 1) \cdot \mathrm{y}_{6,3} \geq 0 \\
 & \mathrm{y}_{6,1} + \mathrm{y}_{6,2} + \mathrm{y}_{6,3} \geq 1 \\
C_7: & \mathrm{b}_{2,1} + \mathrm{b}_{4,1} - (K_1 - 1) \cdot \mathrm{y}_{7,1} \geq 0 \\
 & \mathrm{b}_{5,2} + \mathrm{b}_{10,2} - (K_2 - 1) \cdot \mathrm{y}_{7,2} \geq 0 \\
 & \mathrm{y}_{7,1} + \mathrm{y}_{7,2} \geq 1
\end{array}$$

$$
\begin{array}{lll}
C_8: & b_{2,1} + b_{4,1} - (K_1 - 1) \cdot y_{8,1} & \geq 0 \\
 & b_{7,2} + b_{10,2} - (K_2 - 1) \cdot y_{8,2} & \geq 0 \\
 & y_{8,1} + y_{8,2} & \geq 1 \\
C_9: & b_{5,1} - K_1 \cdot y_{9,1} & \geq 0 \\
 & b_{4,2} - K_2 \cdot y_{9,2} & \geq 0 \\
 & b_{3,3} + b_{8,3} + b_{10,3} - (K_3 - 2) \cdot y_{9,3} & \geq 0 \\
 & y_{9,1} + y_{9,2} + y_{9,3} & \geq 1 \\
C_{10}: & b_{5,1} - K_1 \cdot y_{10,1} & \geq 0 \\
 & b_{4,2} + b_{3,2} + b_{9,2} - (K_2 - 2) \cdot y_{10,2} & \geq 0 \\
 & b_{8,3} + b_{10,3} - (K_3 - 1) \cdot y_{10,3} & \geq 0 \\
 & y_{10,1} + y_{10,2} + y_{10,3} & \geq 1 \\
C_{11}: & b_{5,1} - K_1 \cdot y_{11,1} & \geq 0 \\
 & b_{4,2} + b_{10,2} - (K_2 - 1) \cdot y_{11,2} & \geq 0 \\
 & b_{3,3} - K_3 \cdot y_{11,3} & \geq 0 \\
 & y_{11,1} + y_{11,2} + y_{11,3} & \geq 1 \\
C_{12}: & b_{4,2} + b_{5,2} - (K_2 - 1) \cdot y_{12,2} & \geq 0 \\
 & b_{3,3} + b_{8,3} + b_{10,3} - (K_3 - 2) \cdot y_{12,3} & \geq 0 \\
 & y_{12,2} + y_{12,3} & \geq 1 \\
C_{13}: & b_{3,2} + b_{4,2} + b_{5,2} + b_{9,2} - (K_2 - 3) \cdot y_{13,2} & \geq 0 \\
 & b_{8,3} + b_{10,3} - (K_3 - 1) \cdot y_{13,3} & \geq 0 \\
 & y_{13,2} + y_{13,3} & \geq 1 \\
C_{14}: & b_{4,2} + b_{5,2} + b_{10,2} - (K_2 - 2) \cdot y_{14,2} & \geq 0 \\
 & b_{3,3} - K_3 \cdot y_{14,3} & \geq 0 \\
 & y_{14,2} + y_{14,3} & \geq 1 \\
C_{15}: & b_{10,1} - K_1 \cdot y_{15,1} & \geq 0 \\
 & b_{4,2} - K_2 \cdot y_{15,2} & \geq 0 \\
 & b_{3,3} - K_3 \cdot y_{15,3} & \geq 0 \\
 & y_{15,1} + y_{15,2} + y_{15,3} & \geq 1 \\
C_{16}: & b_{10,1} - K_1 \cdot y_{16,1} & \geq 0 \\
 & b_{4,2} + b_{9,2} + b_{3,2} - (K_2 - 2) \cdot y_{16,2} & \geq 0 \\
 & y_{16,1} + y_{16,2} & \geq 1 \\
C_{17}: & b_{4,2} + b_{7,2} + b_{10,2} - (K_2 - 2) \cdot y_{17,2} & \geq 0 \\
 & b_{3,3} - K_3 \cdot y_{17,3} & \geq 0 \\
 & y_{17,2} + y_{17,3} & \geq 1 \\
C_{18}: & b_{2,1} + b_{4,1} + b_{5,1} - (K_1 - 2) \cdot y_{18,1} & \geq 0 \\
 & b_{8,3} + b_{10,3} - (K_3 - 1) \cdot y_{18,3} & \geq 0 \\
 & y_{18,1} + y_{18,3} & \geq 1 \\
C_{19}: & b_{10,2} - K_2 \cdot y_{19,2} & \geq 0 \\
 & b_{5,3} + b_{6,3} - (K_3 - 1) \cdot y_{19,3} & \geq 0 \\
 & y_{19,2} + y_{19,3} & \geq 1
\end{array}
$$

Φ^*	$K_1 = 10$ $K_2 = 10$ $K_3 = 10$		$K_1 = 46$ $K_2 = 4$ $K_3 = 125$		$K_1 = 3$ $K_2 = 3$ $K_3 = 3$	
	optimal	heuristic	optimal	heuristic	optimal	heuristic
B^*	36	37	175	175	8	8
$b_{2,1}$	9	9	45	45	2	2
$b_{4,2}$	10	10	4	4	3	3
$b_{10,2}$	10	9	4	4	3	3
$b_{6,3}$	7	9	122	122	0	0

Table 4: Results for Example 2.

Note that all y_{l_r} are (0,1)-variables. Table 4 shows the solution of the optimization for different values of $\underline{K} = (K_1, K_2, K_3)$. We included only variables which evaluated nonzero values. The solutions for variables b_{i_r} are an optimal allocation Φ^*. For $\underline{K} = (3,3,3)$ the deadlock conditions for cycle C_3 and C_{13} are satisfied without allocating any capacity (see Corollary 1). Therefore, the constraints in the integer program which were derived from cycles C_3 and C_{13} can be eliminated for the optimization. For all values of $\underline{K}$, the heuristic method from subsection 4.2.2 provides results which are very close to or identical with the optimal solution.

7.3 Example 3

In Figure 5 we show a multiple chain queueing network with a tandem sequence of length $S = 5$. Obtaining an optimal buffer allocation algorithm for networks which include a tandem sequence is a formidable task because of the high number of cycles. However, with the *Reduction Lemma* (Lemma 5) we can replace the tandem sequence, i.e., stations $(6,7,8,9,10)$, by a single station. The reduced network contains only 11 cycles which have to be considered for the optimization:

$$
\begin{aligned}
C_1 &= (B_{1,3}, B_{2,3}, B_{3,3}, B_{1,3}) \\
C_2 &= (B_{1,4}, B_{\{6,7,8,9,10\},4}, B_{1,4}) \\
C_3 &= (B_{4,3}, B_{\{6,7,8,9,10\},3}, B_{4,3}) \\
C_4 &= (B_{4,2}, B_{\{6,7,8,9,10\},2}, B_{4,2}) \\
C_5 &= (B_{5,1}, B_{\{6,7,8,9,10\},1}, B_{5,1}) \\
C_6 &= (B_{5,2}, B_{\{6,7,8,9,10\},1}, B_{5,2}) \\
C_7 &= (B_{4,3}, B_{\{6,7,8,9,10\},2}, B_{4,3}) \\
C_9 &= (B_{1,4}, B_{\{6,7,8,9,10\},3}, B_{1,4}) \\
C_{10} &= (B_{1,3}, B_{4,2}, B_{\{6,7,8,9,10\},4}, B_{1,3}) \\
C_{11} &= (B_{1,3}, B_{4,3}, B_{\{6,7,8,9,10\},4}, B_{1,3})
\end{aligned}
$$

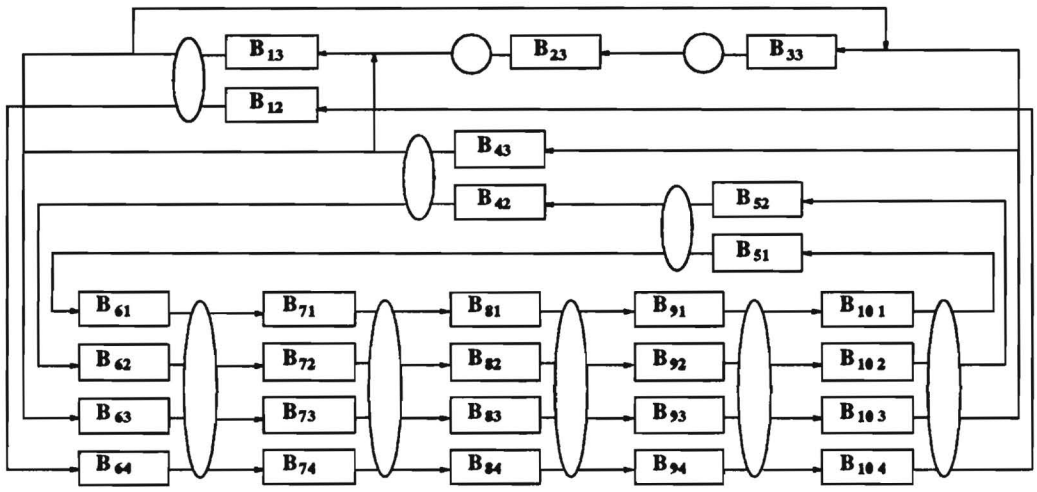


Figure 5: Multiple Chain Queueing Network with a Tandem Sequence.

$B_{\{6,7,8,9,10\},r}$ (for $r = 1,2,3,4$) denotes the chain-r-buffer of the station which replaces the tandem sequence. In Table 5 we present an optimal buffer allocation $\tilde{\Phi}$ for the reduced network with $\underline{K} = (25,30,35,40)$. Again, only nonzero values are shown.

Since the *Reduction Lemma* is only valid if the condition in equation (54) is satisfied, we have to show that the reduction was valid. Since none of the cycles $(C_1, C_2, \ldots, C_{11})$ includes buffers from all routing chains and $K_r > S$ for $r = 1,2,3,4$, condition (54) is clearly satisfied for all routing chains. Therefore, we obtain an optimal allocation for the network in Figure 5 with the results from Table 5 by:

$$\Phi^*(B_{ir}) = \begin{cases} \mathrm{b}_{\{6,7,8,9,10\},r} & \text{if } i = 6 \\ 0 & \text{if } i \in \{7,8,9,10\} \\ \mathrm{b}_{ir} & \text{otherwise} \end{cases}$$

7.4 Example 4

If the network contains more than one tandem sequence, we can apply the *Reduction Lemma* several times. Figure 6 shows a multiple chain queueing network with 4 tandem sequences. The cycle finding algorithm from section 5 generates 1593 cycles for this network. The *Reduction Lemma* allows us to reduce each of the tandem sequences $(1,2,3,4), (5,6,7), (8,9,10)$ and $(11,12,13,14,15)$ to a single station. The reduced network has a structure as shown in Figure 6. We denote the buffers of the network by $\tilde{B}_{ir}$, if B_{ir} is a buffer of the first station in a tandem sequence of the

$\tilde{\Phi}$	$K_1 = 25$ $K_2 = 30$ $K_3 = 35$ $K_4 = 40$	
	optimal	heuristic
$\tilde{B}$	162	163
$b_{\{6,7,8,9,10\},1}$	25	25
$b_{4,2}$	30	30
$b_{3,3}$	33	33
$b_{4,3}$	34	35
$b_{1,4}$	40	40

Table 5: Results for Example 3.

original network. The cycle finding algorithm comes up with only 17 cycles which are relevant for the optimization.

In Table 6 we present an optimal buffer allocation $\tilde{\Phi}$ obtained from solving the optimization corresponding to the network in Figure 7. We solved the network for $\underline{K} = (20, 20, 20)$. Since we applied the *Reduction Lemma* more than once, we have to show that condition (54) holds each time we replace a tandem sequence by a single station. It is a tedious but straightforward task to show that condition (54) is satisfied in each reduction step. We obtain an optimal allocation Φ^* for the original network in Figure 6:

$$\Phi^*(B_{ir}) = \begin{cases} \tilde{b}_{ir} & \text{if } i \in \{1, 5, 8, 11\} \\ 0 & \text{otherwise} \end{cases}$$

$\tilde{\Phi}^*$	$K_1 = 2$ $K_2 = 20$ $K_3 = 6$	
	optimal	heuristic
$\tilde{B}^*$	100	100
$\tilde{b}_{5,1}$	20	20
$\tilde{b}_{5,2}$	20	20
$\tilde{b}_{5,3}$	0	20
$\tilde{b}_{8,1}$	20	0
$\tilde{b}_{8,2}$	20	20
$\tilde{b}_{8,3}$	20	20

Table 6: Results for Example 4.

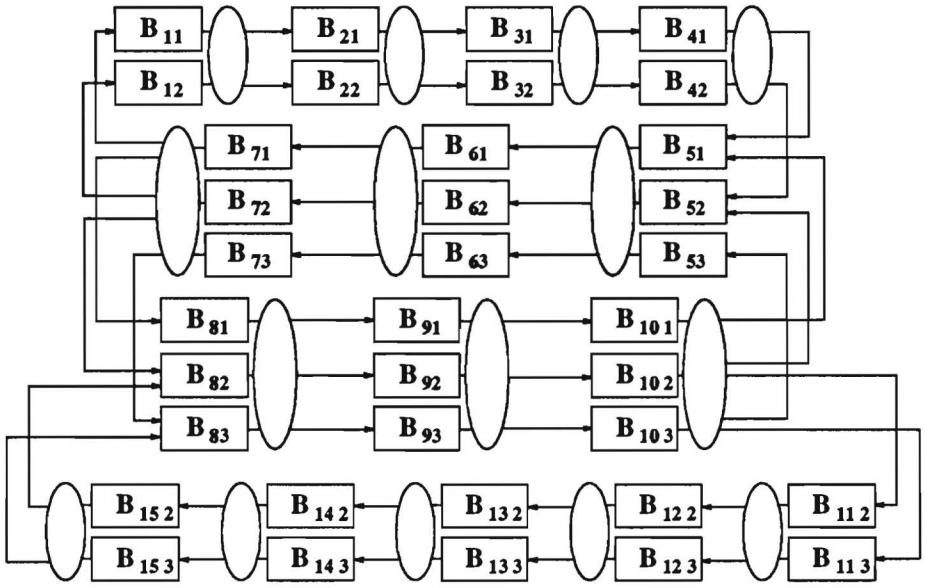


Figure 6: Multiple Chain Queueing Network with Several Tandem Sequences.

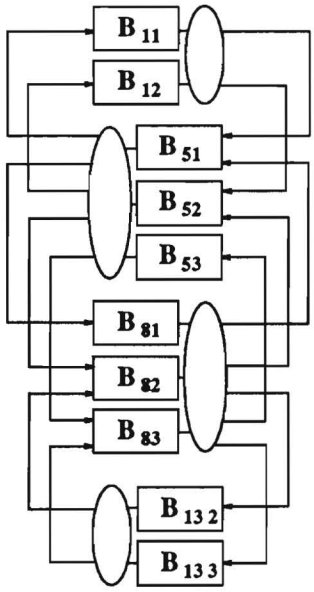


Figure 7: Reduced Queueing Network.

8 Conclusions

We presented the necessary and sufficient conditions for a deadlock free blocking network with multiple routing chains. The correctness of the conditions was proved in section 3. Then, we addressed the problem of finding an optimal deadlock free buffer allocation for a given network. We presented a general algorithm which generates an optimal allocation for multiple chain networks without any restriction on the topology. The algorithm was made efficient by an improved method to find a certain subset of cycles in a directed graph. This method was discussed and evaluated in section 5. In section 6 we showed that for some network topologies we can reduce the effort to find an optimal buffer allocation. Examples in section 7 demonstrated the application of the optimization algorithm.

References

[1] I. F. Akyildiz and H. G. Perros, "Queueing Networks with Finite Capacity Queues", *Special Issue in Performance Evaluation*, Vol. 10, No. 4, December 1989.

[2] S. Kundu and I. F. Akyildiz, "Deadlock Free Buffer Allocation in Closed Queueing Networks", *Queueing Systems*, Vol. 4, No. 1, January 1989, pp. 47 - 56.

[3] D. B. Johnson, "Finding all the Elementary Circuits in a Directed Graph", *SIAM Journal on Computing*, Vol. 4, No. 1, March 1975, pp. 77 - 84.

[4] L. Schrage, "Linear, Integer, and Quadratic Programming with LINDO", *The Scientific Press*, Third Edition, 1986.

Appendix

The following algorithm describes the extended version of *Johnson's algorithm* as discussed in section 5.1.

INPUT: $G = (V, E)$ represented by the adjacency list A_G.
OUTPUT: $\tilde{C}$
COMMENTS: A stack is used with operations POP(), PUSH(), INIT_STACK() and GET_STACK() (write current stack into a list). Set operators are used for list manipulations.

DATA STRUCTURES:

```
int                 N;
boolean             blocked[1..N];
vertex              s;
vertex list array   B[1..N];
vertex list         V_scc;
vertex list array   A_scc[1..N]
list of vertex lists Cycle;
```

```
UPDATE_CYCLE (vertex list newcycle)
  begin
    for (C ∈ Cycle) do
      if (C ⊆ newcycle)
        return;
      else if (newcycle ⊆ C)
        Cycle := Cycle \ {C};
    Cycle := Cycle ∪ {newcycle};
  end
```

```
SIMPLE_CYCLE()
  begin
    for (v ∈ V) do
      if (v ∈ A_G(v))
        begin
          Cycle := Cycle ∪ {v};
          A_G := A_G \ {A_G(v)};
          V := V \ {v};
          for (w ∈ V_G) do
            A_G(w) := A_G(w) \ {v};
        end
    for (v, w ∈ V) do
      if (v ∈ A_G(w) ∧ w ∈ A_G(v)) do
        begin
          UPDATE_CYCLE({v, w});
          A_G(w) := A_G(w) \ {v};
          A_G(v) := A_G(v) \ {w};
        end
  end
```

```
boolean CHECK_STACK(vertex list stack)
    begin
        for (C ∈ Cycle)
            if (C ⊆ GET_STACK())
                return(TRUE);
        return (FALSE);
    end


UNBLOCK(vertex n)
    begin
        vertex w;
        blocked[u] := FALSE;
        for (w ∈ B[u]) do
            begin
                B[u] := B[u] \ {w};
                if (blocked[w])
                    UNBLOCK(w);
            end
    end


boolean CYCLE(vertex v)
    begin
        bool flag;
        flag := FALSE;
        PUSH(v);
        blocked[v] := TRUE;
        if (CHECK_STACK (GET_STACK()))
            flag := TRUE;
        else (w ∈ A_ssc(v)) do
            for (w ∈ A_ssc(v)) do
                if (w = s)
                    begin
                        UPDATE_CYCLE(GET_STACK());
                        flag := TRUE;
                    end
                else if (¬blocked[w])
                    if (CYCLE(w))
                        flag := TRUE;
        if (flag)
            UNBLOCK(v);
        else
            for (w ∈ A_ssc(v)) do
                if (v ∉ B[w])
                    B[w] := B[w] ∪ {w};
        POP();
        return(flag);
    end
```

```
begin
    Cycle := ∅;
    SIMPLE_CYCLE();
    CLEAR_STACK();
    for (v ∈ V) do
        begin
            A_scc := adjacency structure of SCC with
                        least vertex in subgraph of G induced by {n, n + 1, ..., N};
            V_scc := vertex list corresponding to A_scc;
            if (|V_scc| > 2)
                begin
                    s := vertex with least index in V_scc;
                    for (i ∈ V_scc) do
                        begin
                            blocked[i] := FALSE;
                            B[i] := ∅;
                        end
                    CYCLE(s);
                end
        end
    for (C ∈ Cycle) do
        PRINT(C);
end
```

Exact Analysis of Queueing Networks with Multiple Job Classes and Blocking-After-Service *

Ian F. Akyildiz and Chueng-Chiu Huang

School of Electrical Engineering

Georgia Institute of Technology

Atlanta, GA 30332

U.S.A.

December 18, 1992

Abstract

Two models of closed queueing networks with blocking after service and multiple job classes are analyzed. The first model is a network with N stations and each station has either Type II or Type III. The second model is a star like queueing network also called as central server model in which the stations may have either Type I or Type IV, with the condition that the neighbors of these stations must be of Type II or Type III such that blocking will be caused only by this set of station types. Exact product form solutions are obtained for the equilibrium state probabilities in both models. Formulae for performance measures such as throughput and mean number of jobs are also derived.

Key Words: Performance Evaluation, Queueing Networks, Finite Buffer Capacity, Blocking, Deadlock, Equilibrium State Probabilities, Throughput, Mean Number of Jobs.

1 Introduction

In recent years there has been an increased interest in the analysis of queueing networks with blocking. This is probably due to the realization that these queueing networks are useful in modelling

* **This work was supported by National Science Foundation (NSF) under Grant No. CCR-90-11981.**

computer systems, communication networks, and flexible manufacturing systems. The set of rules that dictate when a station becomes blocked and when it becomes unblocked is commonly referred to as the *blocking* mechanism. There are basically only a few blocking mechanisms that have been extensively studied in the literature, Akyildiz and Perros [3] and Onvural [13]. We consider the so-called *blocking after service* Akyildiz and Perros [3] and Onvural [13], (in short form BAS) mechanism, i.e., if a job finishes service at a station and wants to enter a station which is full, it stays in the server of the source station, waiting for a space to be available in the destination station. This blocking policy is also known type 1 blocking, transfer blocking, manufacturing blocking, production blocking and non-immediate blocking in the literature. Several papers consider this blocking policy e.g., Akyildiz [1,2], Akyildiz and von Brand [4], Balsamo et. al. [6], Balsamo and Donatiello [7], Bocharov [9], Onvural [12] and Onvural and Perros [14]. In particular, Onvural and Perros [14] and Akyildiz and von Brand [4] consider closed queueing networks with BAS mechanism for limited number of jobs. Akyildiz and Liebeherr [5] determine necessary and sufficient conditions for deadlock-freedom. Akyildiz and von Brand [4] considered queueing networks with BAS mechanism. In the first model they analyze a two-station model with different types of stations and single class of jobs. They derive an exact product form solution for the equilibrium state probabilities. Akyildiz and von Brand [4] also analyze a second model with N stations, multiple classes of jobs and the limited number of jobs $K = \min_{i \in N} \{B_i\} + 1$, where B_i is the buffer capacity of station i. If the stations that will cause blocking are type I station in $BCMP$ model, an exact product form solution is obtained.

In this paper we consider two models of closed queueing networks with BAS mechanism. In both models we assume that there are N stations and R classes of jobs. The first model is a general topology network with N stations where each station is either of Type II or Type III, Baskett et. al. [8], Kelly [10,11]. The second model is a star like queueing network (also known as a central server model) where the central server must have the station Type I or Type IV [8,10,11] and the neighbors must be either of Type II or Type III stations. In the second model we assume that the blocking will be caused only by the central server. For both models we obtain exact product form solution for equilibrium state probabilities. We also derive formulae for performance measures such as throughput and mean number of jobs.

The paper is organized as follows: In section 2 introduce common notations and definitions for both models. In section 3 we analyze the first model. In section 4 we investigate the central server model. In section 5 we conclude the paper.

2 Basic Model Description

Both models contain N stations, R job classes and K total number of jobs. The types of stations we consider in this paper are from $BCMP$ or $Kelly$ model, Baskett et. al. [8], Kelly [10,11]:

- Type I: The service discipline is first-come-first-served (FCFS); all job classes have the same service time distribution and the service rate can be state-dependent where $\mu(k)$ will denote the service rate with k jobs.

- Type II: There is a single server and the service discipline is processor sharing (PS). Each job class may have a distinct service time distribution.

- Type III: The number of servers is greater or equal to the maximum number of jobs which can be queued at the station; infinite servers (IS). Each job class may have a distinct service time distribution.

- Type IV: There is a single server and the queueing discipline is last-come-first-served preemptive-resume (LCFS-PR). Each job may have a distinct service time distribution.

Here we assume that all service times follow exponential distribution which depend on the station and on the class. Note that our results can easily be extended to general service time distributions for Types II, III and IV. For the sake of simplicity here we give results only for exponential cases.

The transition probabilities are denoted by $p_{ir,js}$, $1 \leq i,j \leq N$ and $1 \leq r,s \leq R$, where a class r job departs from station i and visits station j and becomes a class s job. We also assume that the routing matrix $\mathbf{P} = [p_{i,r,j,s}]$, is irreducible. We define

$$\alpha_{js} = \sum_{i=1}^{N} \sum_{r=1}^{R} \alpha_{ir} p_{ir,js}. \tag{1}$$

Let the vector $\mathbf{n} = (\mathbf{x}_1, \cdots, \mathbf{x}_N)$ denote the number of jobs in N stations and their positions in each station, i.e., $\mathbf{x}_i = (x_{i1}, \cdots, x_{in_i})$, where x_{il} denotes the class of the job in station i at the position l. Let n_i be the number of jobs in station i, and $n_i = n_{i1} + \cdots + n_{iR}$, where n_{ir} is the number of class r jobs in station i. We note that if station i is of Type II or III, then there is no requirement for the order of the jobs, thus, we denote $\mathbf{x}_i$ as $\mathbf{n}_i$, $\mathbf{n}_i = (n_{i1}, \cdots, n_{iR})$.

A product form solution for equilibrium state probabilities exists when there is no blocking in the network, Baskett et. al. [8], Kelly [10,11]:

$$p(\mathbf{n}) = C \prod_{i=1}^{N} f_i(\mathbf{x}_i), \tag{2}$$

where C is the normalization constant such that the sum of equilibrium state probabilities will be equal to one and $f_i(\mathbf{x}_i)$ is defined by the type of station i,

$$f_i(\mathbf{x}_i) = \begin{cases} \left(\frac{1}{\mu_i}\right)^{n_i} \prod_{l=1}^{n_i} \alpha_{ix_{il}}, & \text{if station } i \text{ is Type } I \\ n_i! \prod_{r=1}^{R} \frac{1}{n_{ir}!} \left(\frac{\alpha_{ir}}{\mu_{ir}}\right)^{n_{ir}}, & \text{if station } i \text{ is Type } II \\ \prod_{r=1}^{R} \frac{1}{n_{ir}!} \left(\frac{\alpha_{ir}}{\mu_{ir}}\right)^{n_{ir}}, & \text{if station } i \text{ is Type } III \\ \prod_{l=1}^{n_i} \left(\frac{\alpha_{ix_{il}}}{\mu_{ix_{il}}}\right). & \text{if station } i \text{ is Type } IV \end{cases} \tag{3}$$

We define T_1 as the set of Type I or IV stations, and T_2 as the set of Type II or III stations. Since we are considering the models with blocking, we need to define B_i as the capacity of station i (i.e., buffer and server capacity). In addition to avoid the self loop that cause *deadlock* we assume that

$$p_{ii} = 0, \text{ for } i = 1, \cdots, N.$$

For convenience we define the following quantities for each station type. For type II and III station, the service rate are dependent on the number of jobs in the station and on their classes but independent of their order. Thus, we let $\mathbf{n}_i = (n_{i1}, \cdots, n_{iR})$ for $i \in T_2$, where n_{ir} denotes the number of class r active jobs in station i. A job is said to be active if it is not blocked. Also we let $n_i = n_{i1} + \cdots + n_{iR}$ be the total number of active jobs in station i for $i \in T_2$.

Let b denote the number of blocked jobs. Thus, with the active jobs we have

$$K = \sum_{i=1}^{N} n_i + b, \tag{4}$$

with

$$0 \leq b \leq K - \min_{1 \leq i \leq N} \{B_i\}. \tag{5}$$

We assume that a job will choose its destination station and the class when it finishes its service. Considering the b blocked jobs we need to know their locations and their classes. We also assume that the blocked jobs will join that station, which caused blocking according to $First - Come - First - In$ $(FCFI)$ scheduling discipline. Thus, we let $\mathbf{y} = (\mathbf{y}_i)$ and $\mathbf{y}_i = (y_{i1}, \cdots, y_{ib_i})$ if $b_i \geq 1$ otherwise $\mathbf{y}_i = \emptyset$ where b_i denotes the number of jobs blocked by station i, and y_{il}, $1 \leq y_{il} \leq N$ denotes the location of the lth job blocked by station i. Similarly, $\mathbf{z} = (\mathbf{z}_i)$ and $\mathbf{z}_i = (z_{i1}, \cdots, z_{ib_i})$ where z_{il}, $1 \leq z_{il} \leq R$ denotes the class of the lth blocked job. To have the complete information about the system we define the state space $S = \{(\mathbf{n}, \mathbf{x}, \mathbf{y}, \mathbf{z})\}$, where $\mathbf{x} = (\mathbf{x}_i, i \in T_1)$ and $\mathbf{n} = (\mathbf{n}_i, i \in T_2)$. For a given state $(\mathbf{n}, \mathbf{x}, \mathbf{y}, \mathbf{z})$ the service rate of station i for $i \in T_2$ is $\mu_i(\mathbf{n}_i) = \mu_{i1}(n_{i1}) + \cdots + \mu_{iR}(n_{iR})$ where $\mu_{ir}(n_{ir}) = n_{ir}\mu_{ir}$ if station i is of Type III, and if station i is of Type II, then $\mu_{ir}(n_{ir}) = \frac{n_{ir}}{n_i}\mu_{ir}$. For $i \in T_1$ then the service rate is: if station i is of type I, then $\mu_i(\mathbf{x}_i) = \mu_i$ for $n_i > 0$, otherwise it is equal to 0. If station i is of Type IV, then $\mu_i(\mathbf{x}_i) = \mu_{in_i}$ for $n_i \leq B_i$.

Let e_{ir} denote an NxR matrix with all zero elements except the element at the position of ith row and rth column with value 1.

3 General Topology Network with Type II and III Stations

In this section we consider a general topology network with only Type II and III stations. We assume the following constraint in the model:

$$K < \min_{1 \leq i,j \leq N; p_{ij} > 0} \{B_i + B_j\}, \tag{6}$$

This constraint says that only one of the N stations may cause blocking at any time. The reason for making this assumption is to avoid more than two jobs moving at one event. Thus, the state space is reduced to $S = (\mathbf{n}, \mathbf{y}, \mathbf{z})$, where $\mathbf{y}$ and $\mathbf{z}$ are b by 1 vectors, because if there are any blocked jobs, then they are all blocked by the same station.

Now we consider the transitions of the jobs for a nonblocking state $(\mathbf{n}, \mathbf{y}, \mathbf{z})$, where $\mathbf{y}$ and $\mathbf{z}$ are empty vectors $\emptyset$. In this case the system will reach the state $(\mathbf{n} - e_{ir} + e_{js}, \emptyset, \emptyset)$ with rate $\mu_{ir}(n_{ir}) p_{ir,js}$, if $n_j \neq B_j$. Otherwise, a class r job from station i will be blocked by station j and the transition will occur from state $(\mathbf{n}, \emptyset, \emptyset)$ to state $\left(\mathbf{n} - e_{ir}, \mathbf{y}', , \mathbf{z}'\right)$ where $\mathbf{y}' = (y_1)$ and $y_1 = i$. The same occurs for $\mathbf{z}' = (z_1)$ and $z_1 = s$. The rate to reach this state is the same as before, i.e., $\mu_{ir}(n_{ir}) p_{ir,js}$.

Now we consider the transitions from a blocking state. Suppose station k is full, i.e., $n_k = B_k$ and the current state is $(\mathbf{n}, \mathbf{y}, \mathbf{z})$, for $b \geq 0$. Then a class r job leaves station i and moves to station j, for $i \neq k, j \neq k$ and becomes a class s job and the system reaches the state $(\mathbf{n} - e_{ir} + e_{js}, \mathbf{y}, \mathbf{z})$ with rate $\mu_{ir}(n_{ir}) p_{ir,js}$. If a class r job moves from station i to station k and becomes a class s job, then the state $(\mathbf{n} - e_{ir}, \mathbf{y} + y_{b+1}, \mathbf{z} + z_{b+1})$ and $y_{b+1} = i$, $z_{b+1} = s$ will be reached. Note that in this case the transition rate is $\mu_{ir}(n_{ir}) p_{ir,js}$.

Let $\mathbf{y} = (y_1, \cdots, y_b)$ then the operator $\mathbf{y} + y_{b+1}$ will be $(y_1, \cdots, y_{b+1})$. Similar is also valid for the operator $\mathbf{z} + z_{b+1}$. When considering a class r job which moves from station k to station j and becomes a class s job with the transition rate $\mu_{kr}(n_{kr}) p_{kr,js}$, then one of the first blocked jobs will move to station k at the same time. Thus, the new state is $\left(\mathbf{n} - e_{kr} + e_{js} + e_{kz}, \mathbf{y}', \mathbf{z}'\right)$, with $\mathbf{y}' = \left(y_1', \cdots y_{b-1}'\right)$, $y_l' = y_{l+1}$ and $\mathbf{z}' = \left(z_1', \cdots z_{b-1}'\right)$, $z_l' = z_{l+1}$ for $1 \leq l \leq b-1$.

¿From these state transitions we obtain the global balance equation for a state $(\mathbf{n}, \mathbf{y}, \mathbf{z}) \in S$ if $n_i < B_i$ for $1 \leq i \leq N$, thus $\mathbf{y} = \emptyset$ and $\mathbf{z} = \emptyset$:

$$p(\mathbf{n}, \emptyset, \emptyset) \sum_{i=1}^{N} \sum_{r=1}^{R} \mu_{ir}(n_{ir}) = \sum_{i=1}^{N} \sum_{r=1}^{R} \sum_{j=1}^{N} \sum_{s=1}^{R} p(\mathbf{n} + e_{ir} - e_{js}, \emptyset, \emptyset) \mu_{ir}(n_{ir} + 1) p_{ir,js}, \tag{7}$$

and the global balance equation for a blocking state (suppose $n_k = B_k$):

$$
\begin{aligned}
& p(\mathbf{n},\mathbf{y},\mathbf{z}) \sum_{i=1}^{N}\sum_{r=1}^{R} \mu_{ir}(n_{ir}) \\
&= \sum_{i=1,i\neq k}^{N}\sum_{r=1}^{R}\sum_{j=1,j\neq k}^{N}\sum_{s=1}^{R} p(\mathbf{n}+e_{ir}-e_{js},\mathbf{y},\mathbf{z})\,\mu_{ir}(n_{ir}+1)\,p_{ir,js} \\
&+ \sum_{j\neq k}\sum_{s=1}^{R}\sum_{r=1}^{R}\sum_{y_0=1}^{N}\sum_{z_0=1}^{R} p(\mathbf{n}+e_{kr}-e_{js}-e_{kz_0},\mathbf{y}+y_0,\mathbf{z}+z_0)\,\mu_{kr}(n_{kr})\,p_{kr,js} \\
&+ \sum_{s=1}^{R} p(\mathbf{n}+e_{y_b s},\mathbf{y}-y_b,\mathbf{z}-z_b)\,\mu_{y_b s}(n_{y_b s}+1)\,p_{y_b s,kz_b}. \qquad (8)
\end{aligned}
$$

The first term on the right hand side of (8) is the total flow-in rate to the state $(\mathbf{n},\mathbf{y},\mathbf{z})$ from state $(\mathbf{n}+e_{ir}-e_{js},\mathbf{y},\mathbf{z})$. We note that the moving job is from station i to station j, $i\neq k, j\neq k$. The second term on the right-hand side of (8) is the flow-in rate to state $(\mathbf{n},\mathbf{y},\mathbf{z})$ by a class r job moving out from station k, and in the mean time another blocked job, which is class z_0, will move into station k from station y_0. As a consequence a transition occurs from the state $(\mathbf{n}+e_{kr}-e_{js}-e_{kz_0},\mathbf{y}+y_0,\mathbf{z}+z_0)$ to the state $(\mathbf{n},\mathbf{y},\mathbf{z})$.

For notational convenience we let y_0, z_0 denote the location and the class of the blocked job that will move into station k while a job leaves station k and joins station i. The third term in (8) is the flow-in rate to the state $(\mathbf{n},\mathbf{y},\mathbf{z})$ by a joining job at station k.

We note that a given state will be visited and the departure job is from a station i, $i\neq k$, then we know where the job comes from, i.e., $y_b=i$ and its class is switched from s to z_b. So that the transition is from the state $(\mathbf{n}+e_{y_b s},\mathbf{y}-y_b,\mathbf{z}-z_b)$ to the state $(\mathbf{n},\mathbf{y},\mathbf{z})$.

Theorem 1. The model has the following product form solution for equilibrium state probabilities

$$
p(\mathbf{n},\mathbf{y},\mathbf{z}) = C\prod_{i=1}^{N} f_i(\mathbf{n}_i)\left(\prod_{l=1}^{b}\frac{\sum_{t=1}^{R}\alpha_{y_l t}p_{y_l t,iz_l}}{\sum_{t=1}^{R}\mu_{it}(n_{it})}\right)^{I_{\{n_i=B_i\}}} \qquad (9)
$$

where $f_i(\mathbf{n}_i)$ is defined

$$
f_i(\mathbf{n}_i) = \left\{ \begin{array}{ll} n_i!\prod_{r=1}^{R}\frac{1}{n_{ir}!}\left(\frac{\alpha_{ir}}{\mu_{ir}}\right)^{n_{ir}} & \text{if station } i \text{ is type } II \\ \prod_{r=1}^{R}\frac{1}{n_{ir}!}\left(\frac{\alpha_{ir}}{\mu_{ir}}\right)^{n_{ir}} & \text{if station } i \text{ is type } III \end{array} \right\}. \qquad (10)
$$

and C is a normalization constant

$$
C^{-1} = \sum_{(\mathbf{n},\mathbf{y},\mathbf{z})\in S}\prod_{i=1}^{N} f_i(\mathbf{n}_i)\left(\prod_{l=1}^{b}\frac{\sum_{t=1}^{R}\alpha_{y_l t}p_{y_l t,iz_l}}{\sum_{t=1}^{R}\mu_{it}(n_{it})}\right)^{I_{\{n_i=B_i\}}} \qquad (11)
$$

Proof. We escape the proof of the nonblocking states since it is exactly the same as in the *BCMP* or *Kelly* model, Baskett et. al. [8] and Kelly [10,11]. For the global balance equation of

blocking states we assume that the state k has $n_k = B_k$ and the number of blocked jobs is b, then for (9) and (10) we have

$$p\left(\mathbf{n}+e_{ir}-e_{js},\mathbf{y},\mathbf{z}\right)=p\left(\mathbf{n},\mathbf{y},\mathbf{z}\right)\frac{\alpha_{ir}}{\mu_{ir}\left(n_{ir}+1\right)}\frac{\mu_{js}\left(n_{js}\right)}{\alpha_{js}} \tag{12}$$

if $i \neq k$ and $j \neq k$.

Next we consider the departure event of a job leaving station $j \neq k$ and visiting station k

$$\begin{aligned} &p\left(\mathbf{n}+e_{kr}-e_{js}-e_{kz_0},\mathbf{y}+y_0,\mathbf{z}+z_0\right) \\ &=p\left(\mathbf{n},\mathbf{y},\mathbf{z}\right)\frac{\alpha_{kr}}{\mu_{kr}(n_{kr}+1)}\frac{\mu_{js}(n_{js})}{\alpha_{js}}\frac{\mu_{kz_0}(n_{kz_0})}{\alpha_{kz_0}}\frac{\sum_{t=1}^{R}\alpha_{y_0t}p_{y_0t,kz_0}}{\sum_{t=1}^{R}\mu_{kt}(n_{kt})} \end{aligned} \tag{13}$$

and for the departure event of a job leaving station k and visiting station i

$$p\left(\mathbf{n}+e_{y_bs},\mathbf{y}-y_b,\mathbf{z}-z_b\right)=p\left(\mathbf{n},\mathbf{y},\mathbf{z}\right)\frac{\alpha_{y_bs}}{\mu_{y_bs}\left(n_{y_bs}+1\right)}\frac{\sum_{t=1}^{R}\mu_{kt}\left(n_{kt}\right)}{\sum_{t=1}^{R}\alpha_{y_bt}p_{y_bt,kz_b}}. \tag{14}$$

By substituting (12) - (14) to the right-hand side of (8) separately and using (1), we obtain

$$\begin{aligned} &\sum_{i=1,i\neq k}^{N}\sum_{r=1}^{R}\sum_{j=1,j\neq k}^{N}\sum_{s=1}^{R}p\left(\mathbf{n}+e_{ir}-e_{js},\mathbf{y},\mathbf{z}\right)\mu_{ir}\left(n_{ir}+1\right)p_{ir,js} \\ &=p\left(\mathbf{n},\mathbf{y},\mathbf{z}\right)\sum_{j=1,j\neq k}^{N}\sum_{s=1}^{R}\frac{\mu_{js}(n_{js})}{\alpha_{js}}\left[\sum_{i=1,i\neq k}^{N}\sum_{r=1}^{R}\alpha_{ir}p_{ir,js}\right] \end{aligned} \tag{15}$$

$$\begin{aligned} &\sum_{j\neq k}\sum_{s=1}^{R}\sum_{r=1}^{R}\sum_{y_0=1}^{N}\sum_{z_0=1}^{R}p\left(\mathbf{n}+e_{kr}-e_{js}-e_{kz_0},\mathbf{y}+y_0,\mathbf{z}+z_0\right)\mu_{kr}\left(n_{kr}\right)p_{kr,js} \\ &=\ p\left(\mathbf{n},\mathbf{y},\mathbf{z}\right)\sum_{j\neq k}\sum_{s=1}^{R}\sum_{r=1}^{R}\sum_{z_0=1}^{R}\frac{\mu_{js}\left(n_{js}\right)}{\alpha_{js}}\frac{\alpha_{kr}}{\alpha_{kz_0}}\frac{\mu_{kz_0}\left(n_{kz_0}\right)}{\sum_{t=1}^{R}\mu_{kt}\left(n_{kt}\right)}p_{kr,js}\left[\sum_{y_0=1}^{N}\sum_{t=1}^{R}\alpha_{y_0t}p_{y_0t,kz_0}\right] \\ &=\ p\left(\mathbf{n},\mathbf{y},\mathbf{z}\right)\sum_{j\neq k}\sum_{s=1}^{R}\frac{\mu_{js}\left(n_{js}\right)}{\alpha_{js}}\sum_{r=1}^{R}\left[\alpha_{kr}p_{kr,js}\sum_{z_0=1}^{R}\left(\frac{\mu_{kz_0}\left(n_{kz_0}\right)}{\sum_{t=1}^{R}\mu_{kt}\left(n_{kt}\right)}\right)\right] \\ &=\ p\left(\mathbf{n},\mathbf{y},\mathbf{z}\right)\sum_{j\neq k}\sum_{s=1}^{R}\frac{\mu_{js}\left(n_{js}\right)}{\alpha_{js}}\left[\sum_{r=1}^{R}\alpha_{kr}p_{kr,js}\right] \end{aligned} \tag{16}$$

By combining (15) and (16) we have

$$\begin{aligned} (15)+(16) &= p\left(\mathbf{n},\mathbf{y},\mathbf{z}\right)\sum_{j\neq k}\sum_{s=1}^{R}\frac{\mu_{js}\left(n_{js}\right)}{\alpha_{js}}\left[\sum_{i=1,i\neq k}^{N}\sum_{r=1}^{R}\alpha_{ir}p_{ir,js}+\sum_{r=1}^{R}\alpha_{kr}p_{kr,js}\right] \\ &= p\left(\mathbf{n},\mathbf{y},\mathbf{z}\right)\sum_{j\neq k}\sum_{s=1}^{R}\frac{\mu_{js}\left(n_{js}\right)}{\alpha_{js}}\left[\sum_{i=1}^{N}\sum_{r=1}^{R}\alpha_{ir}p_{ir,js}\right] \\ &= p\left(\mathbf{n},\mathbf{y},\mathbf{z}\right)\sum_{j\neq k}\sum_{s=1}^{R}\mu_{js}\left(n_{js}\right). \end{aligned} \tag{17}$$

In (14) we substitute the third term of (8) and obtain

$$\begin{aligned}
&\sum_{s=1}^{R} p\left(\mathbf{n}+e_{y_b s}, \mathbf{y}-y_b, \mathbf{z}-z_b\right) \mu_{y_b s}\left(n_{y_b s}+1\right) p_{y_b s, k z_b} \\
&= p(\mathbf{n}, \mathbf{y}, \mathbf{z}) \frac{\sum_{t=1}^{R} \mu_{kt}\left(n_{kt}\right)}{\sum_{t=1}^{R} \alpha_{y_b t} p_{y_b t, k z_b}} \sum_{s=1}^{R} \alpha_{y_b s} p_{y_b s, k z_b} \\
&= p(\mathbf{n}, \mathbf{y}, \mathbf{z}) \sum_{t=1}^{R} \mu_{kt}\left(n_{kt}\right). \qquad (18)
\end{aligned}$$

Thus, with (17) and (18) we have $p(\mathbf{n}, \mathbf{y}) \sum_{j=1}^{N} \sum_{s=1}^{R} \mu_{js}\left(n_{js}\right)$ which is equal to the right-hand side of (8). **QED**

Remark. The result (9) follows from the following argument: we consider the buffers occupied by the blocked jobs which are stored in Type II or III stations only, as the extended buffers of the stations causing blocking. The existence of blocked jobs in Type II or Type III stations will not effect the service of the non-blocking jobs in these stations. That is the throughput rate of these two types of stations will be dependent on the number of non-blocked jobs in this station only. Thus, we can ignore the locations of the blocked jobs and the result (9) will follow.

Now we derive the throughput of the system $\lambda = \sum_{i=1}^{N} \lambda_i$, where λ_i is the throughput of station i, and the mean number of jobs $\overline{k}_i$ in station i.

Theorem 2. The *throughput* of station i is computed from

$$\begin{aligned}
\lambda_i = \sum_{(\mathbf{n},\mathbf{y},\mathbf{z}) \in S} p(\mathbf{n}, \mathbf{y}, \mathbf{z}) \Bigg[&\sum_{r=1}^{R} \mu_{ir}\left(n_{ir}\right) \left(1 - \sum_{k=1}^{N} \sum_{s=1}^{R} p_{ir,ks} I_{\{n_k = B_k\}}\right) \\
&+ \sum_{k=1}^{N} I_{\{n_k = B_k\}} I_{\{y_1 = i\}} \left(\sum_{r=1}^{R} \mu_{kr}\left(n_{kr}\right)\right)\Bigg] \qquad (19)
\end{aligned}$$

The *mean number of jobs* at station i is

$$\overline{k}_i = \sum_{(\mathbf{n},\mathbf{y},\mathbf{z}) \in S} p(\mathbf{n}, \mathbf{y}, \mathbf{z}) \left(\sum_{r=1}^{R} n_{ir} + \sum_{l=1}^{b} I_{\{y_l = i\}}\right). \qquad (20)$$

Proof. For any given state $(\mathbf{n}, \mathbf{y}, \mathbf{z})$ we consider the throughput of station i. Suppose the number of class r active jobs is n_{ir}, thus, the service rate is $\mu_{ir}\left(n_{ir}\right)$. However, a job which completed its service in station i can leave station i only by choosing a nonfull station j, i.e., $n_j < B_j$ with probability $\sum_{s=1}^{R} p_{ir,js}$. When considering the blocked jobs, if the first blocked job in station i is blocked by station k, i.e., $y_1 = i$, then this job has the same rate with which will leave station i and join the station k as the service rate of station k, i.e., $\sum_{r=1}^{R} \mu_{kr}\left(n_{kr}\right)$. Thus, (19) follows.

Similarly, for given a state $(\mathbf{n}, \mathbf{y}, \mathbf{z})$ we have the number of jobs in station i which will be equal to the active jobs $\sum_{r=1}^{R} n_{ir}$ plus the blocked jobs in this station, i.e., $\sum_{l=1}^{b} I_{\{y_l = i\}}$, so that (20) follows. **QED**

4 Central Server Model with Different Types of Stations

In this section we analyze central server models (star networks) with BAS mechanism. We put the constraint that the central server must be a station of Type either I or IV and all other stations may be of Type II or III. We further assume that only stations of Type I or IV will cause blocking, and the neighbors of such stations must be of Type II or III. The total number of jobs must satisfy the constraint

$$\min_{i\in T_1}\{B_i\} < K < \min_{i\in T_2}\{B_i\}$$

and $p_{ir,js} = 0$, if $i, j \in T_1$.

We note that $p_{ir,js} = 0$ if $i, j \in T_1$ and for all r, s, then (1) will become:

$$\alpha_{js} = \sum_{i\in T_2}\sum_{r=1}^{R} \alpha_{ir} p_{ir,js} \qquad for \quad j \in T_1 \tag{21}$$

We denote T_A as the set of stations in T_1 if $n_i < B_i$, and T_B as the set of stations in T_1 with $n_i = B_i$. Note that $T_A \cup T_B = T_1$. In the following we give the global balance equation for a given state $(\mathbf{n}, \mathbf{x}, \mathbf{y}, \mathbf{z})$.

$$\begin{aligned}
&p(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z})\left(\sum_{i\in T_1}\mu_i(\mathbf{x}_i) + \sum_{i\in T_2}\sum_{r=1}^{R}\mu_{ir}(n_{ir})\right) \qquad (22)\\
&= \sum_{j\in T_A}\sum_{i\in T_2}\sum_{r=1}^{R} p\left(\mathbf{n}+e_{ir}, \mathbf{x}-x_{jn_j}, \mathbf{y}, \mathbf{z}\right)\mu_{ir}(n_{ir}+1)\, p_{ir,jx_{jn_j}}\\
&+ \sum_{j\in T_B}\sum_{r=1}^{R} p\left(\mathbf{n}+e_{y_{jb_j}r}, \mathbf{x}, \mathbf{y}-y_{jb_j}, \mathbf{z}-z_{jb_j}\right)\mu_{y_{jb_j}r}\left(n_{y_{jb_j}r}+1\right) p_{y_{j,b_j}r,jz_{jb_j}}\\
&+ \sum_{i\in T_2, j\subset nT_2}\sum_{s=1}^{R}\sum_{r=1}^{R} p(\mathbf{n}+e_{ir}-e_{js}, \mathbf{x}, \mathbf{y}, \mathbf{z})\,\mu_{ir}(n_{ir}+1)\, p_{ir,js}\\
&+ \sum_{i\in T_A, j\in T_2}\sum_{s=1}^{R}\sum_{x_{i0}=1}^{R} p(\mathbf{n}-e_{js}, \mathbf{x}+x_{i0}, \mathbf{y}, \mathbf{z})\,\mu_i(\mathbf{x}_i+x_{i0})\, p_{ix_{i0},js}\\
&+ \sum_{i\in T_B, j\in T_2}\sum_{s=1}^{R}\sum_{y_{i0}=1}^{N}\sum_{x_{i0}=1}^{R} p(\mathbf{n}-e_{js}, \mathbf{x}+x_{i0}-x_{iB_i}, \mathbf{y}+y_{i0}, \mathbf{z}+z_{i0})\,\mu_i(\mathbf{x}_i+x_{i0}-x_{iB_i})\, p_{ix_{i0},js}.
\end{aligned}$$

We consider a given state $(\mathbf{n}, \mathbf{x}, \mathbf{y}, \mathbf{z})$ then the total flow-out rate from this state is $p(\mathbf{n}, \mathbf{x}, \mathbf{y}, \mathbf{z})\left(\sum_{i\in T_1}\mu_i(\mathbf{x}_i) + \sum_{i\in T_2}\sum_{r=1}^{R}\mu_{ir}(n_{ir})\right)$. In order to check the flow-in rates we consider the different events caused by these jobs: The first term on the right hand side of (22) denotes the departure event of a class r job from station $i \in T_2$ to station $j \in T_A$ to class x_{jn_j}. The second term denotes the events of a class r job in station $y_{jb_j} \in T_2$ which completed its service and is joining station $j \in T_B$. However, since station j is full, $n_j = B_j$, this job is blocked and becomes a class z_{jb_j} job and joins the list of blocked jobs waiting for station j. The third term denotes the event of moving

of a class r job from station $i \in T_2$ to station $j \in T_2$ and becoming a class s job. The last two terms describing the events of moving of a class x_{i0} job from station $i \in T_A$, $i \in T_B$ to station $j \in T_2$ and becoming a class s job. We note that x_{i0} denotes the class in which the job leaves station i. y_{i0} and z_{i0} denote the station and the class of the blocked job at the first position of the list of blocked jobs which will move to station i. Note that $z_{i0} = x_{iB_i}$.

Theorem 3. The model has the following product form solution for equilibrium state probabilities:

$$p(\mathbf{n}, \mathbf{x}, \mathbf{y}, \mathbf{z}) = C \prod_{i \in T_1} f_i(\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i) \prod_{i \in T_2} f_i(\mathbf{n}_i), \tag{23}$$

where $f_i(\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i)$ for $i \in T_1$ is defined as

$$f_i(\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i) = \begin{cases} \left(\frac{1}{\mu_i}\right)^{n_i} \prod_{l=1}^{n_i} \alpha_{ix_{il}}, & \text{if } i \text{ is Type } I \text{ and } i \in T_A \\ \left(\frac{1}{\mu_i}\right)^{n_i} \prod_{l=1}^{B_i} \alpha_{ix_{il}} \prod_{l=1}^{b_i} \frac{\sum_{t=1}^{R} \alpha_{y_{il}t} p_{y_{il}t,iz_{il}}}{\mu_i}, & \text{if } i \text{ is Type } I \text{ and } i \in T_B \\ \prod_{l=1}^{n_i} \frac{\alpha_{ix_{il}}}{\mu_{ix_{il}}}, & \text{if } i \text{ is Type } IV \text{ and } i \in T_A \\ \prod_{l=1}^{B_i} \frac{\alpha_{ix_{il}}}{\mu_{ix_{il}}} \prod_{l=1}^{b_i} \frac{\sum_{t=1}^{R} \alpha_{y_{il}t} p_{y_{il}t,iz_{il}}}{\mu_{iz_{il}}}, & \text{if } i \text{ is Type } IV \text{ and } i \in T_B \end{cases} \tag{24}$$

and $f_i(\mathbf{n}_i)$ for $i \in T_2$, is defined as

$$f_i(\mathbf{n}_i) = \begin{cases} n_i! \prod_{r=1}^{R} \frac{1}{n_{ir}!} \left(\frac{\alpha_{ir}}{\mu_{ir}}\right)^{n_{ir}} & \text{if station } i \text{ is Type } II \\ \prod_{r=1}^{R} \frac{1}{n_{ir}!} \left(\frac{\alpha_{ir}}{\mu_{ir}}\right)^{n_{ir}} & \text{if station } i \text{ is Type } III \end{cases} \tag{25}$$

where C is a normalization constant

$$C^{-1} = \sum_{(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z}) \in S} \prod_{i \in T_1} f_i(\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i) \prod_{i \in T_2} f_i(\mathbf{n}_i). \tag{26}$$

Proof. From (23) - (25) we have

$$p\left(\mathbf{n} + e_{ir}, \mathbf{x} - x_{jn_j}, \mathbf{y}, \mathbf{z}\right) = p(\mathbf{n}, \mathbf{x}, \mathbf{y}, \mathbf{z}) \frac{\mu_j(\mathbf{x}_j)}{\alpha_{jx_{jn_j}}} \frac{\alpha_{ir}}{\mu_{ir}(n_{ir}+1)} \tag{27}$$

$$\begin{aligned} &p\left(\mathbf{n} + e_{y_{jb_j}r}, \mathbf{x} - x_{jn_j}, \mathbf{y} - y_{jb_j}, \mathbf{z} - z_{jb_j}\right) \\ &= p(\mathbf{n}, \mathbf{x}, \mathbf{y}, \mathbf{z}) \frac{\mu_j(\mathbf{x}_j)}{\sum_{t=1}^{R} \alpha_{y_{jb_j}t} p_{y_{jb_j}t,jz_{jb_j}}} \frac{\alpha_{y_{jb_j}r}}{\mu_{y_{jb_j}r}\left(n_{y_{jb_j}r}+1\right)} \end{aligned} \tag{28}$$

$$p(\mathbf{n} + e_{ir} - e_{js}, \mathbf{x}, \mathbf{y}, \mathbf{z}) = p(\mathbf{n}, \mathbf{x}, \mathbf{y}, \mathbf{z}) \frac{\alpha_{ir}}{\mu_{ir(n_{ir}+1)}} \frac{\mu_{js(n_{js})}}{\alpha_{js}} \tag{29}$$

$$p\left(\mathbf{n}-e_{js},\mathbf{x}+x_{i0},\mathbf{y},\mathbf{z}\right)=p\left(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z}\right)\frac{\alpha_{ix_{i0}}}{\mu_i\left(\mathbf{x}_i+x_{i0}\right)}\frac{\mu_{js(n_{js})}}{\alpha_{js}} \tag{30}$$

$$\begin{aligned}&p\left(\mathbf{n}-e_{js},\mathbf{x}+x_{i0}-x_{iB_i},\mathbf{y}+y_{i0},\mathbf{z}+z_{i0}\right)\\&=\ p\left(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z}\right)\frac{\mu_{js(n_{js})}}{\alpha_{js}}\frac{\alpha_{ix_{i0}}}{\mu_i\left(\mathbf{x}_i+x_{i0}-x_{iB_i}\right)}\frac{\sum_{t=1}^{R}\alpha_{y_{i0}t}p_{y_{i0}t,ix_{B_i}}}{\alpha_{ix_{iB_i}}}\frac{\mu_i\left(\mathbf{x}_i\right)}{\mu_i\left(\mathbf{z}_i+z_{i0}\right)}\end{aligned} \tag{31}$$

Note that the last term on the right hand side of (31) is $\frac{\mu_i(\mathbf{x}_i)}{\mu_i(\mathbf{z}_i+z_{i0})}=1$, because if the station i is of Type I, then the service rate is independent of the class. However, if the station is of Type IV then $\mu_i\left(\mathbf{x}_i\right)=\mu_{ix_{iB_i}}$. Since we assume the blocked jobs are $FCFI$, then $\mu_i\left(\mathbf{z}_i+z_{i0}\right)=\mu_{iz_{i0}}$ and $x_{iB_i}=z_{i0}$. Now we substitute (27) – (31) on the right-hand side of (22) separately, and use (1) to obtain

$$\begin{aligned}&\sum_{j\in T_A}\sum_{i\in T_2}\sum_{r=1}^{R}p\left(\mathbf{n}+e_{ir},\mathbf{x}-x_{jn_j},\mathbf{y},\mathbf{z}\right)\mu_{ir}\left(n_{ir}+1\right)p_{ir,jx_{jn_j}}\\&=\ p\left(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z}\right)\sum_{j\in T_A}\sum_{i\in T_2}\sum_{r=1}^{R}\frac{\mu_j\left(\mathbf{x}_j\right)}{\alpha_{jx_{jn_j}}}\alpha_{ir}p_{ir,jx_{jn_j}}\\&=\ p\left(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z}\right)\sum_{j\in T_A}\mu_j\left(\mathbf{x}_j\right)\end{aligned} \tag{32}$$

$$\begin{aligned}&\sum_{j\in T_B}\sum_{r=1}^{R}p\left(\mathbf{n}+e_{y_{jb_j}r},\mathbf{x},\mathbf{y}-y_{jb_j},\mathbf{z}-z_{jb_j}\right)\mu_{y_{jb_j}r}\left(n_{y_{jb_j}r}+1\right)p_{y_{j,b_j}r,jz_{jb_j}}\\&=\ p\left(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z}\right)\sum_{j\in T_B}\sum_{r=1}^{R}\frac{\mu_j\left(\mathbf{x}_j\right)}{\sum_{t=1}^{R}\alpha_{y_{jb_j}t}p_{y_{jb_j}t,jz_{jb_j}}}\alpha_{y_{jb_j}r}p_{y_{j,b_j}r,jz_{jb_j}}\\&=\ p\left(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z}\right)\sum_{j\in T_B}\mu_j\left(\mathbf{x}_j\right)\end{aligned} \tag{33}$$

$$\begin{aligned}&\sum_{i\in T_2,j\in T_2}\sum_{s=1}^{R}\sum_{r=1}^{R}p\left(\mathbf{n}+e_{ir}-e_{js},\mathbf{x},\mathbf{y},\mathbf{z}\right)\mu_{ir}\left(n_{ir}+1\right)p_{ir,js}\\&=\ p\left(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z}\right)\sum_{j\in T_2}\sum_{s=1}^{R}\frac{\mu_{js(n_{js})}}{\alpha_{js}}\sum_{i\in T_2}\sum_{r=1}^{R}\alpha_{ir}p_{ir,js}\end{aligned} \tag{34}$$

$$\begin{aligned}&\sum_{i\in T_A,j\in T_2}\sum_{s=1}^{R}\sum_{x_{i0}=1}^{R}p\left(\mathbf{n}-e_{js},\mathbf{x}+x_{i0},\mathbf{y},\mathbf{z}\right)\mu_i\left(\mathbf{x}_i+x_{i0}\right)p_{ix_{i0},js}\\&=\ p\left(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z}\right)\sum_{j\in T_2}\sum_{s=1}^{R}\frac{\mu_{js(n_{js})}}{\alpha_{js}}\sum_{i\in T_A}\sum_{x_{i0}=1}^{R}\alpha_{ix_{i0}}p_{ix_{i0},js}\end{aligned} \tag{35}$$

$$\sum_{i\in T_B, j\in T_2}\sum_{s=1}^{R}\sum_{y_{i0}=1}^{N}\sum_{x_{i0}=1}^{R} p\left(\mathbf{n}-e_{js},\mathbf{x}+x_{i0}-x_{iB_i},\mathbf{y}+y_{i0},\mathbf{z}+z_{i0}\right)\mu_i\left(\mathbf{x}_i+x_{i0}-x_{iB_i}\right)p_{ix_{i0},js}$$

$$= p\left(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z}\right)\sum_{i\in T_B, j\in T_2}\sum_{s=1}^{R}\sum_{x_{i0}=1}^{R}\frac{\mu_{js(n_{js})}}{\alpha_{js}}\frac{\alpha_{ix_{i0}}p_{ix_{i0},js}}{\alpha_{ix_{iB_i}}}\sum_{y_{i0}=1}^{N}\sum_{t=1}^{R}\alpha_{y_{i0}t}p_{y_{i0}t,ix_{iB_i}}$$

$$= p\left(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z}\right)\sum_{j\in T_2}\sum_{s=1}^{R}\frac{\mu_{js(n_{js})}}{\alpha_{js}}\left(\sum_{i\in T_B}\sum_{x_{i0}=1}^{R}\alpha_{ix_{i0}}p_{ix_{i0},js}\right) \tag{36}$$

By combining (34) – (36) we have

$$(34)+(35)+(36) = p\left(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z}\right)\sum_{j\in T_2}\sum_{s=1}^{R}\frac{\mu_{js(n_{js})}}{\alpha_{js}}\left(\sum_{i=1}^{N}\sum_{r=1}^{R}\alpha_{ir}p_{ir,js}\right)$$

$$= p\left(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z}\right)\sum_{j\in T_2}\sum_{s=1}^{R}\mu_{js(n_{js})}. \tag{37}$$

The result follows by checking the flow-in rate equal to the flow-out rate. **QED**

Remark. The result (23) is obtained by similar arguments as in (9). The blocked jobs are kept in an extended buffer at a station of Type II or Type III. The blocked jobs behavior is captured through the state description in our model.

Theorem 4.

The *throughput* of the ith station is computed from

$$\lambda_i = \sum_{(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z})\in S} p\left(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z}\right)\begin{cases}\mu_i\left(\mathbf{x}_i\right) & \text{for } i\in T_1\\ \sum_{r=1}^{R}\mu_{ir}\left(r_{ir}\right)\left(1-\sum_{j\in T_B}\sum_{s=1}^{R}p_{ir,js}\right) & \text{for } i\in T_2\\ \quad +\sum_{j\in T_B}\mu_j\left(\mathbf{x}_j\right)I_{\{y_{j1}=i\}}. & \end{cases} \tag{38}$$

The *mean number of jobs* in the ith station is

$$\overline{k}_i = \sum_{(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z})\in S} p\left(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z}\right)\begin{cases} n_i & \text{for } i\in T_1\\ \sum_{r=1}^{R}n_{ir}+\sum_{j\in T_B}\sum_{l=1}^{b_j}I_{\{y_{jl}=i\}} & \text{for } i\in T_2\ . \end{cases} \tag{39}$$

Proof. Consider a state $(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z})$. Since the jobs from stations of T_1 can never be blocked, thus the throughput of such station $i\in T_1$ is $\mu_i\left(\mathbf{x}_i\right)$. For stations in T_2 we have the throughput rate as the service rate $\sum_{r=1}^{R}\mu_{ir}\left(n_{ir}\right)$. However, the job may choose the next station in T_B and a blocking may occur. If station $i\in T_2$ has blocked jobs which has the first priority to get into the

destination station that caused their blocking, then the throughput rate of those jobs from station i is equal to the service rate of the station which caused blocking. Thus (38) follows.

The *mean number of jobs* is also obtained from given a state $(\mathbf{n}, \mathbf{x}, \mathbf{y}, \mathbf{z})$. The number of jobs in station i for $i \in T_1$ is n_i. If $i \in T_2$, then the number of jobs in station i will include the active jobs of all classes $n_{i1} + \cdots + n_{iR}$ and the blocked jobs in that station. Thus, we take sum over the stations that cause blocking, $j \in T_B$, and check the vectors $\mathbf{y}_j$ if $y_{jl} = i$, then the lth job is located in station i, so that the result (38) follows. **QED**

5 Conclusions

We derived exact product form solution for the equilibrium state probabilities and computed throughput and mean number of jobs in two different models. We used a property of Type II and III stations that the locations of the blocked jobs has no effect on the service rates in those stations. Thus, we used this type of stations as kind of *optional* storage spaces for the blocking station. In other words, the system is treated as a virtual *nonblocking* system. By the well-defined state space, particularly the vector of the locations for the blocked jobs, we obtain the exact product form solutions. Another observation is that by the construction of the models we always have the constraint that no more than two jobs will move on a single departure event. The first model has a stronger constraint that is only one of the stations will cause blocking at a time. However, for the second model we allow more than one station to be of Type I or IV which may cause blocking at the same time. This is due to the constraint of the neighbors which will not cause blocking.

REFERENCES

[1] I. F. Akyildiz, "Exact Product Form Solution for Queueing Networks with Blocking", *IEEE Transactions on Computers*, Vol. C-36, No. 1, January 1987, pp. 122-125.

[2] I. F. Akyildiz, "On the Exact and Approximate Throughput Analysis of Closed Queueing Networks with Blocking", *IEEE Transactions on Software Engineering*, Vol. SE-14, No. 1, January 1988, pp. 62-71.

[3] I. F. Akyildiz and H. G. Perros, "Special Issue on Queueing Networks with Finite Capacity Queues: Introduction", *Performance Evaluation*, Vol. 10, No. 3, Dec. 1989.

[4] I. F. Akyildiz and H. von Brand, "Exact Solutions for Networks of Queues with Blocking-After-Service", *Technical Report, Georgia Tech, College of Computing, April 1991, GIT-COC-91-26.*

[5] I. F. Akyildiz and J. Liebeherr, "Optimal Deadlock Free Buffer Allocation in Multiple Chain Blocking Networks of Queues", *Proc. of the Int. Conference on the Performance of Distributed Systems and Integrated Communication Networks*, September 10-12, 1991.

[6] S. Balsamo, V. De Nitto Persone and G. Iazeoalla, "Identity and Reducibility Properties of Some Blocking and Non-blocking Mechanisms in Congested Networks", *Flow Control of Congested Networks*, NATO ASI Series, Springer Verlag, 1987, pp. 243-254.

[7] S. Balsamo and L. Donatiello, "On the Cycle Time Distribution in a Two-Stage Cyclic Network with Blocking", *IEEE Transactions on Software Engineering*, Vol. 15, No. 10, October 1989.

[8] F. Baskett, K. M. Chandy, R. R. Muntz and F. G. Palacios, "Open, Closed and Mixed Networks of Queues with Different Classes of Customers", *Journal of the ACM*, Vol. 22, Vol. 15, pp. 248-260, April 1975.

[9] P. P. Bocharov, "On the Two-Node Queueing Networks with Finite Capacity", *Proc. of the First Int. Workshop on Queueing Networks with Blocking*, North Holland, Dec. 1989, pp. 105-125.

[10] F. P. Kelly, "Networks of Queues with Customers of Different Types", *Journal of Applied Probability*, Vol. 12, pp. 542-554, 1975.

[11] F. P. Kelly, "Networks of Queues", *Advances in Applied Probability*, Vol. 8, No. 2, pp. 416-432, June 1976.

[12] R. O. Onvural, "A Note on the Product Form Solutions of Multiclass Closed Queueing Networks with Blocking", *Performance Evaluation*, Vol. 10, No. 3, December 1989, pp. 247-255.

[13] R. O. Onvural, "A Survey of Closed Queueing Networks with Finite Buffers", *ACM Computing Surveys*, Vol. 22, No. 2, June 1990, pp. 83-121.

[14] R. O. Onvural and H. G. Perros, "Some Equivalencies between Closed Queueing Networks with Blocking", *Performance Evaluation Journal*, Vol. 9, Dec. 1988, pp. 111-118.

EXACT SOLUTIONS FOR NETWORKS OF QUEUES WITH BLOCKING-AFTER-SERVICE

Ian F. Akyildiz and Horst von Brand*

School of Electrical Engineering

Georgia Institute of Technology

Atlanta, GA 30332

U.S.A.

February 23, 1993

*This work was supported by National Science Foundation (NSF) under Grant No. CCR-90-11981.
Permanent address of H. von Brand: Departamento de Informática, Universidad Técnica Federico Santa María, Casilla 110-V, Valparaíso, CHILE.

Abstract

The paper has two major parts. The first part deals with two-station networks with BAS mechanism and different station types. In this part only single class of jobs is allowed. The contribution here is to show that exact solutions exist for two station queueing networks with BAS mechanism having different station types. The exact equilibrium state probability distributions are derived. Insensitivity is investigated and and formulas for performance measures are obtained. It is demonstrated that the throughputs, mean number of jobs and mean number of blocked jobs depend on the scheduling discipline. A queueing network model with more than two stations is analyzed in the second part. Multiple job classes with job class change, and different station types are allowed in the model. Exact solutions for equilibrium state probabilities and performance measures are obtained under the condition that only a certain total number of jobs is allowed in the network.

1 Introduction

Queueing networks have been used increasingly as tools for performance evaluation of computer systems, manufacturing systems and communication networks. For some special cases, which we call ***classical networks***, the exact probability distributions are known [9,15] and efficient algorithms [13] can be used to compute performance measures like throughputs and mean number of jobs.

Classical networks assume that the capacity of the stations offering service is infinite, an assumption that usually does not hold. This gives rise to *queueing networks with blocking*. In recent years there has been an increased interest in the analysis of queueing networks with blocking. This is probably due to the realization that these queueing networks are useful in modeling computer systems, communication networks, and flexible manufacturing systems. Recently a special issue [5] appeared in a journal which give the state-of-the-art in this research area. The set of rules that dictate when a node becomes blocked and when it becomes unblocked is commonly referred to as the ***blocking*** mechanism. There are basically only a few blocking mechanisms that have been extensively studied in the literature [5]. The blocking type we investigate in this work is called type 1 blocking, transfer blocking, production blocking and non-immediate blocking. Recently the naming of blocking mechanisms has been standardized [5], where this case received the name *blocking after service*, (henceforth in short form BAS) where a job upon service completion at station i attempts to join destination station j. If station j at that moment is full, the job is forced to wait in the server of station i, until it enters destination station j. The server remains blocked for this period of time and it cannot serve any other job waiting in the station.

There are very few exact results for systems with BAS. A two-station model was considered by Akyildiz [1] for FCFS service only. Akyildiz shows that there is a nonblocking queueing network with the same transition matrix as the blocking network after relabeling of states, so the model is solved exactly. Balsamo and Donatiello [2] showed exact solutions for cycle time distributions for two-node closed queueing networks with blocking. The basic ideas are taken from [1] a non-blocking queueing network can be found with appropriate total number of jobs which has a product form solution. Onvural and Perros [20] consider closed queueing networks with more than two stations and BAS for limited number of jobs. Onvural [17] discusses product form solutions of several models with blocking-after-service. A recent survey of the area of queueing networks with blocking is given by Onvural [18].

Our literature study reveals that existing or proposed methods either contain disadvantages

(e.g., long run times or memory space) or restrictions (only two-node or tandem network solutions) or provide approximate results which differ widely from the exact values. The queueing network models with BAS mechanism in case of closed queueing networks have the additional limitations such as that all service time distributions are exponential, the queueing discipline at each node is basically FCFS and all jobs belong to the same class. Most of the models of the existing systems show these characteristics. Therefore, in this paper we attack these limitations and obtain exact solutions for certain queueing networks with BAS mechanism. The paper has essentially two major parts. The first part deals with two-station networks with BAS mechanism and different station types. In this part we allow only a single class of jobs. The contribution here is to show exact solutions do exist for two station queueing networks with BAS mechanism having different station types. We were not able to extend these results to multiple job class case. In the second part we allow multiple job classes and different station types. We also permit more than two stations in the network. However, we introduce a condition on the number of permitted jobs in the network so that we can get exact solutions for the model. The paper is organized as follows: In section 2 we analyze the two-station model. First we describe the model in detail. Then we obtain the exact equilibrium state probability distributions. We also show the insensitivity results and obtain formulas for performance measures such as throughput and mean number of jobs. In section 3 we investigate the multiple station, multiple job class queueing networks with BAS mechanism. We describe the model first. We state the condition on the total number of jobs permitted in the network. Then we use the equivalency between blocking and nonblocking networks and obtain exact solution for equilibrium state probabilities. We also derive formulas for performance measures to compute throughput and mean queue lengths. Finally section 4 concludes the paper.

2 Two-Station BAS Networks with Different Station Types and Single Job Class

2.1 Model Description

We consider two stations, labeled 1 and 2. The stations form a cyclic network, i.e. jobs that leave station 1 go to station 2 and viceversa. We will later show that the case in which jobs are allowed to return to the station where they finish service can be reduced to this case, so there is no loss of generality in this. The capacity of station i is denoted by b_i. The service requirements at station i are exponentially distributed with rate μ_i.

We assume a *scheduling discipline* that can be described in terms of the work of [11,12,16], i.e. there are functions:

$f_i(k)$: The total service effort of the server if there are k jobs in station i. Clearly, $f_i(k) = 0$ if $k = 0$.

$\phi_i(l,k)$: The fraction of the service effort expended on the job in position l in the server when there are k jobs in station i. This requires $\sum_{1 \le l \le k} \phi_i(l,k) = 1$ and $\phi_i(l,k) = 0$ if $l < 0$ or $l > k$.

$\psi_i(l,k)$: The probability that a newly arrived job is put in position l of the queue of station i when there are k jobs in the queue before its arrival. This requires $\sum_{1 \le l \le k+1} \psi_i(l,k) = 1$ and $\psi_i(l,k) = 0$ whenever $l < 0$ or $l > k+1$.

A scheduling discipline of this class is called *symmetric* if we also have:

$$\phi(l,k) = \psi(l,k+1) \tag{1}$$

It is a well-known result [9,16] that with scheduling disciplines as these classical (nonblocking) queueing networks have a product form solution whenever either:

- The service time distributions are arbitrary (they may even depend on the job class) and the scheduling discipline is symmetric.
- The service times of all classes have the same exponential distribution.

For the multiple server case we write m_i for the number of places in station i which receive service. For later convenience we define:

$$n_i = \min(k_i, m_i) \tag{2}$$

In terms of the above functions, we have:

$$\phi_i(l,k) = \begin{cases} 0 & \text{if } k = 0 \text{ or } l > n_i \\ 1/n_i & \text{if } 1 \le l \le n_i \end{cases} \tag{3}$$

The blocked jobs in service positions do not make any use of the service effort assigned to their positions in the queue so that service effort is lost.

Note that we have both the total service effort made at the station in the form of $f_i(k_i)$ and also the fraction of this service effort assigned to position l by $\phi_i(l,k_i)$. We get the three cases we consider in this paper by taking m_i as 1 for FCFS and as b_i for PS. The definitions of MS and PS are slightly unusual in that we will assume that for the three cases we compare the functions $f_i(\cdot)$ are the same. Otherwise the comparison would make little sense.

We restrict the functions $\phi(\cdot,\cdot)$ considered to those that distribute the service effort evenly among the served positions of the queue, because in that way we can forget the detailed positions of the blocked jobs in the queue. Otherwise the rates at which jobs finish service at a station is affected by the ordering of the jobs. In this way they only contribute via a binomial coefficient for their multiplicity.

As for the placement function, we assume:

$$\psi_i(l,k) = 0 \qquad \text{for } 1 \le l \le n_i \tag{4}$$

This is to ensure that no newly arrived job can displace a blocked job from a served position in the queue. We will only make use of this fact, so the placement of the arriving jobs could even depend on the blocked jobs present, as long as no blocked job leaves a served position.

The total number of jobs in the network is K. There must be some free space in the network, since otherwise the network deadlocks immediately. We also require that there be blocking, so that:

$$\min(b_1, b_2) < K < b_1 + b_2 \tag{5}$$

2.2 Global Balance Equations

We derive the exact distributions for the three scheduling disciplines, FCFS, PS, and MS. We are mainly interested in the mean number of jobs in each station, the mean number of blocked jobs (which we take to average the mean number of jobs that are in service positions with their service finished, but which cannot proceed since the destination is full) and the throughputs.

We can distinguish three different ranges of operation of this system:

- None of the stations is full, there is no blocking.
- Station 1 is full, there might be blocked jobs in station 2.
- Station 2 is full, there could be blocked jobs in station 1.

Note that two stations cannot be blocked simultanously otherwise deadlock would occur. We assume that the network is deadlock free [6]. Clearly, the situations in which one of the stations contains blocked jobs are symmetric, so it is enough to consider one of them. We will discuss the case in which station 2 is full and station 1 may contain blocked jobs.

The state in the most general case (multiple servers) can be described by

$$(k_1, \kappa_1; k_2, \kappa_2) \tag{6}$$

where k_i is the total number of jobs and κ_i the number of blocked jobs in station i, respectively. This description is redundant, since we have $k_2 = K - k_1$ and κ_i can only be nonzero when $k_{3-i} = b_{3-i}$. We will use because it reflects the underlying symmetry very well.

The restrictions on the possible values of the κ_i are the only difference among the policies considered. For FCFS κ_i is either 0 or 1, for MS we have $0 \leq \kappa_i \leq \min(k_i, m_i)$ and for PS $0 \leq \kappa_i \leq k_i$. In general, we have $0 \leq \kappa_i \leq n_i$, with the definition (2) of n_i.

2.2.1 Balance Equations for Nonblocked States

For nonblocking states the global balance equations are just:

$$\begin{aligned}(\mu_1 f_1(k_1) + \mu_2 f_2(k_2))\pi(k_1, 0; k_2, 0) \\ = \quad \mu_1 f_1(k_1 + 1)\pi(k_1 + 1, 0; k_2 - 1, 0) + \\ \mu_2 f_2(k_2 + 1)\pi(k_1 - 1, 0; k_2 + 1, 0)\end{aligned} \tag{7}$$

Writing this equation in terms of

$$g(k) = \pi(k, 0; K - k, 0) \tag{8}$$

we get:

$$(\mu_1 f_1(k) + \mu_2 f_2(K - k))g(k) = \mu_1 f_1(k + 1)g(k + 1) + \mu_2 f_2(K - k + 1)g(k - 1) \tag{9}$$

holds for $b_1 > k > K - b_2$

Rearranging we have:

$$\mu_2 f_2(K-k)g(k) - \mu_1 f_1(k+1)g(k+1) = \mu_2 f_2(K-k+1)g(k-1) - \mu_1 f_1(k)g(k) \quad (10)$$

Note that both sides of (10) are the expression

$$\mu_2 f_2(K-k)g(k) - \mu_1 f_1(k+1)g(k+1) \quad (11)$$

evaluated for k and for $k-1$, respectively. So this has to be a constant, call it c_1. This gives a linear difference equation for $g(\cdot)$:

$$\mu_2 f_2(K-k)g(k) - \mu_1 f_1(k+1)g(k+1) = c_1 \quad (12)$$

The homogeneous equation (i.e. for $c_1 = 0$) has the general solution:

$$g(k) = c_2 \prod_{1 \le l \le k} \frac{1}{\mu_1 f_1(l)} \prod_{1 \le l \le K-k} \frac{1}{\mu_2 f_2(l)} \quad (13)$$

If this were a network without blocking, we could use the balance equations for $k = 0$ to show that $c_1 = 0$. But in our network the state $k = 0$ is not feasible because of the condition (5). We will assume for now that $c_1 = 0$, a proof of this will be given later. Note that the distribution (13) does not depend on m_i. We will make use of this fact later.

2.2.2 Balance Equations for Blocked States

By the obvious symmetry of the case in which station 1 and station 2 are full, the other one possibly containing blocked jobs, it is enough to consider one of them in detail. We will consider station 2 to be full. Note that this means that station 2 does not change, since as long as station 1 contains blocked jobs, whenever a job finishes service in station 2 it moves into station 1, and a blocked job moves into its place. By the same token, the total number of jobs in station 1 does not change.

A blocked job just wastes its share of the service effort. If κ jobs are blocked, the rate at which jobs leave the station is given by:

$$\mu_i f_i(k_i)(n_i - \kappa_i)/n_i \quad (14)$$

This is a consequence of the form of $\phi(\cdot)$ we assumed in (3). Note that in the blocked states the number of jobs in both stations is constant, the only variable here is κ_i. Also, as only jobs that are being served can get blocked, $0 \le \kappa_i \le n_i$, where n_i is defined by (2).

The states we are considering here are of the form $(K - b_2, \kappa_1; b_2, 0)$. To make the equations more compact, we will use k_1 as a shorthand for $K - b_2$ in some of the equations that follow. The only free variable here is κ_1. So we write the balance equations in terms of:

$$h_1(\kappa) = \pi(K - b_2, \kappa; b_2, 0) \quad (15)$$

This gives:

$$(\mu_1 f_1(k_1)\frac{n_1-\kappa}{n_1} + \mu_2 f_2(b_2))h_1(\kappa)$$
$$= \mu_1 f_1(k_1)\frac{n_1-\kappa+1}{n_1}h_1(\kappa-1) + \mu_2 f_2(b_2)h_1(\kappa+1) \quad (16)$$

Here the first terms refer to transitions in which a job finishes service in station 1 and gets blocked; while the second terms are for jobs that finish service at station 2, so they move into station 1 and a blocked job from station 1 becomes unblocked and moves into station 2. Again, the method used to derive (10) from (9) can be applied, which gives:

$$\mu_2 f_2(b_2)h_1(\kappa+1) - \mu_1 f_1(k_1)\frac{n_1-\kappa}{n_1}h_1(\kappa) = c_3 \quad (17)$$

To get c_3 we use the balance equation for the case when $\kappa = n_1$, i.e. all jobs inside station 1 are blocked. The balance equation for that case is:

$$\mu_1 f_1(k_1)\frac{1}{n_1}h_1(n_1-1) = \mu_2 f_2(b_2)h_1(n_1) \quad (18)$$

This is exactly equation (17) for $\kappa = n_1$ with $c_3 = 0$. So we have the solution:

$$h_1(\kappa) = \frac{c_{14}}{\kappa!}\left(\frac{1}{\mu_1 f_1(k_1)}\right)^{n_1-\kappa}\left(\frac{1}{\mu_2 f_2(b_2)}\right)^{\kappa} \quad (19)$$

Obviously this only makes sense for $0 \leq \kappa \leq n_1$.

2.3 Exact Solution for Equilibrium State Probabilities

There remains the problem of determining the value of the constant c_1. For this we use the balance equations for the state in which station 2 is full but no jobs are blocked inside station 1, since this case is covered both by (10) and (17), for the cases when $k = K - b_2$ and $\kappa = 0$, respectively. The value of $g(K-b_2)$ is the same as the value of $h_1(0)$, since both are the probabilities of the same physical situation.

For the special case $k = K - b_2 - 1$, equation (10) is:

$$\mu_2 f_2(b_2-1)g(k_1+1) - \mu_1 f_1(k_1)g(k_1) = c_1 \quad (20)$$

The balance around the state in which station 2 is full, but there are no blocked jobs inside station 1 is:

$$(\mu_1 f_1(k_1) + \mu_2 f_2(b_2))\, g(k_1) = \mu_2 f_2(b_2)h_1(1) + \mu_1 f_1(k_1+1)g(k_1+1) \quad (21)$$

By the solution (19) we have the following relation between $h_1(0)$ and $h_1(1)$:

$$h_1(1) = h_1(0)\frac{\mu_1 f_1(k_1)}{\mu_2 f_2(b_2)} \quad (22)$$

which together with (21) and the equality $g(k_1) = h_1(0)$ noted above gives:

$$(\mu_1 f_1(k_1) + \mu_2 f_2(b_2))\, g(k_1) = \mu_1 f_1(k_1) g(k_1) + \mu_1 f_1(k_1+1) g(k_1+1) \tag{23}$$

which in turn reduces to:

$$\mu_2 f_2(b_2) g(k_1) = \mu_1 f_1(k_1+1) g(k_1+1) \tag{24}$$

which is exactly equation (20)(with $c_1 = 0$. This proves the claim made in section (3.2), so (13) is really the solution we are after.

The (three-piece) solution we have now involves three arbitrary constants, namely c_2, c_{14} and c_{24}, which corresponds to c_{14} but for station 2.

Since they are used very often in what follows, we define the constants N_1 and N_2 by:

$$N_1 = \min(m_1, K - b_2) \tag{25}$$

$$N_2 = \min(m_2, K - b_1) \tag{26}$$

This are the values of n_1 and n_2 for blocked stations.

Making use of the fact that $g(k_1) = h_1(0)$, we can find the relationship between the two constants c_2 and c_{14} using equations (13) and (19). Equating both expressions and simplifying, we have:

$$c_2 \prod_{1 \le l \le K-b_2} \frac{1}{\mu_1 f_1(l)} \prod_{1 \le l \le b_2} \frac{1}{\mu_2 f_2(l)} = c_{14} \left(\frac{1}{\mu_1 f_1(K-b_2)} \right)^{N_1} \tag{27}$$

On the other hand, repeating the same procedure for station 2 will give us another constant c_{24}, for which we again have the equivalent of (27):

$$c_2 \prod_{1 \le l \le b_1} \frac{1}{\mu_1 f_1(l)} \prod_{1 \le l \le K-b_1} \frac{1}{\mu_2 f_2(l)} = c_{24} \left(\frac{1}{\mu_2 f_2(K-b_1)} \right)^{N_2} \tag{28}$$

The single remaining independent constant is determined by the fact that the sum of the probabilities of all states has to be one.

Pulling together the results of sections (3.2) and (3.3), and using the value of c_2 given by (27) the complete distribution is given by:

$$\pi(k_1, \kappa_1; k_2, \kappa_2) = \begin{cases} c_2 \prod_{1 \le l \le k} \frac{1}{\mu_1 f_1(l)} \prod_{1 \le l \le K-k} \frac{1}{\mu_2 f_2(l)} \\ \frac{c_{14}}{\kappa_1!} \left(\frac{1}{\mu_1 f_1(K-b_2)} \right)^{N_1 - \kappa_1} \left(\frac{1}{\mu_2 f_2(b_2)} \right)^{\kappa_1} \\ \frac{c_{24}}{\kappa_2!} \left(\frac{1}{\mu_1 f_1(b_1)} \right)^{\kappa_2} \left(\frac{1}{\mu_2 f_2(K-b_1)} \right)^{N_2 - \kappa_2} \end{cases} \tag{29}$$

The first case is for nonblocked states, i.e. states of the form $(k_1, 0; k_2, 0)$; the second case is for states in which jobs in station 1 are being blocked by station 2, i.e. of the form $(K - b_2, \kappa_1; b_2, 0)$ with $0 \le \kappa_1 \le N_1$; and the third case is the symmetrical of the second, states of the form $(b_1, 0; K - b_1, \kappa_2)$ with $0 \le \kappa_2 \le N_2$. Note that the "blocked" states with $\kappa = 0$ are really the same as the states in which the stations are full.

2.4 Recirculation

The case considered up to here is the case of a cyclic network, where jobs go to station 2 after finishing service at station 1 and viceversa. In general, recirculation (jobs that return to their origin station immediately) cannot be allowed because it can give rise to deadlock. But in the case where the "recirculating" station has infinite capacity, no deadlock can arise. This is the case when, for example

$$b_2 < K < b_1 \tag{30}$$

In this case jobs can never get blocked inside station 2, since station 1 can never be full. We can then assume that the probability that a job leaving station 1 wants to return to station 1 is nonzero. Calling this value α, the complete set of values for the p_{ij} is as follows:

$$\begin{pmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{pmatrix} = \begin{pmatrix} \alpha & 1-\alpha \\ 1 & 0 \end{pmatrix} \tag{31}$$

Note that when a nonblocked job finishes service at station 1 and returns to it, the state of the network does not really change. The balance equations (7) and (16) are then modified only in that the rate at which jobs leave station 1 is multiplied by $(1 - \alpha)$. So, the network with recirculation around station 1 is governed by the same balance equations as the network we have been considering, just with

$$\hat{\mu}_1 = (1-\alpha)\mu_1 \tag{32}$$

Also, the throughput of station 1 increases:

$$\hat{\lambda}_1 = \frac{\lambda_1}{1-\alpha} \tag{33}$$

Except for that, the performance measures of the network with recirculation are the same as in the network without, just with the corrected μ_1.

2.5 Insensitivity

One interesting property of classical networks is the so-called *insensitivity*, which means that the probabilities of the states (and consequently of the performance measures) do not depend on the scheduling discipline nor on the distribution of the service time distributions when the network has a product form solution. For *repetitive-service-blocking*, where a job that cannot enter a full station returns to its original station and gets another round of service there, insensitivity has been proved by van Dijk and Tijms [23] for two station networks like ours. The results of Akyildiz and von Brand [3,4] also imply insensitivity for this repetitive-service-blocking mechanism as long as routing is reversible, as it is for cyclic two station networks. So there is some hope of getting insensitivity in this case also. Besides, based on our extensive simulations, we conjectured that there is insensitivity for networks with BAS mechanism when different scheduling disciplines are used. However, the distributions for this simple case are different. We will now show that the mean number of jobs and throughput depend on the scheduling discipline in our model.

To simplify the derivations, we consider the symmetric case in which:

$$\mu_1 = \mu_2 \tag{34}$$
$$b_1 = b_2 \tag{35}$$

We will furthermore assume that:

$$f_1(l) = f_2(l) = \begin{cases} 0 & \text{if } l = 0 \\ 1 & \text{if } l > 0 \end{cases} \tag{36}$$

From now on we will drop the subscripts on these quantities.

2.5.1 Equilibrium State Probabilities

For this case the distribution (13) takes a particularly simple form. We will continue to use the functions $g(\cdot)$ and $h_i(\cdot)$ defined by (8) and (15), respectively. We have:

$$c_{14} = c_2 \mu^{N_1 - K} \tag{37}$$
$$c_{24} = c_2 \mu^{N_2 - K} \tag{38}$$

so the distribution turns out to be:

$$g(k) = c_2 \mu^{-K} \tag{39}$$
$$h_i(\kappa) = \frac{c_2 \mu^{-K}}{\kappa!} \tag{40}$$

The constant c_2 is determined from:

$$1 = \sum_{K-b \le k \le b} g(k) + \sum_{1 \le \kappa_1 \le N_1} h_1(\kappa_1) + \sum_{1 \le \kappa_2 \le N_2} h_2(\kappa_2) \tag{41}$$

The cases $\kappa = 0$ are omitted in the second and third sums since in those states no job is blocked, and so this case is included in the first summation. Substituting (39) and (40) into (41) we get:

$$1 = c_2 \left((b - (K - b) + 1)\mu^{-K} + \mu^{-K} S(N_1) + \mu^{-K} S(N_2) \right) \tag{42}$$

Here we used the shorthand:

$$S(N) = \sum_{1 \le k \le N} \frac{1}{k!} \tag{43}$$

Note that this is close to the sum that defines e:

$$e = \sum_{0 \le k < \infty} \frac{1}{k!} \tag{44}$$

This sum converges very rapidly, as is clear from the factorial in the denominator.

Rearranging (42) we have:

$$c_2 = \mu^K (2b - K + 1 + S(N_1) + S(N_2)) \tag{45}$$

2.5.2 Mean Number of Jobs

Now we are ready to compute the mean number of jobs in each station. The mean number of jobs inside station i is:

$$\bar{k}_i = c_2 \mu^{-K} \left(\sum_{K-b \le k_i \le b} k_i + (K-b)S(N_1) + bS(N_2) \right) \tag{46}$$

The second term is the probability that there are blocked jobs inside station 1, in which case there are $K - b$ jobs in it. Similarly, the third term is for the case where there are blocked jobs inside station 2, in which case station 1 is full.

It is clear from (46) that if the values of the N_i are different for the stations, the mean number of jobs in them will be different. But, given our symmetry assumptions, the only difference between the stations is the scheduling discipline. Note also that the difference will normally be very small, since it is essentially a part of the sum in (44).

2.5.3 Throughput

Even though the mean number of jobs is affected by the scheduling discipline, it is conceivable that the throughput is not. We now show that the throughput also depends on the scheduling discipline. As the throughput is exactly the same in both stations, we will consider the case in which $N_1 = N_2$, in addition to the symmetry requirements we imposed in equations (34), (35) and (36). We will then show that throughput depends on their common value.

The throughput is essentially the mean number of service positions that are active (i.e. occupied by a job that is receiving service) in any one of the stations, say station 1. This is given by:

$$\lambda = \mu^{-K+1} c_2 \left(\sum_{K-b \le k \le m} k + \sum_{m < k \le b} m + \sum_{1 \le \kappa \le K-b} \frac{K-b-\kappa}{\kappa!} + mS(m) \right) \tag{47}$$

Here the first term corresponds to the case where not all service positions in the station are occupied, while the second term is still for the case where the station itself is not full (or is full but does not block any jobs) but all the service positions are occupied, were we are arbitrarily assuming that the number of service positions is greater than the minimal number of jobs in the station. The third term is for the case in which blocked jobs occupy service positions and the fourth term considers the case in which the station is full, blocking jobs in the other station.

Rearranging equation (47) and expressing the third term in terms of $S(\cdot)$ we have:

$$\lambda = c_2 \mu^{-K+1} \left(\frac{(2b+1)m - m^2}{2} + mS(m) - \frac{(K-b-1)(K-b)}{2} \right.$$

$$+ (K-b)S(K-b) - S(K-b-1) - 1\Big) \tag{48}$$

Clearly this depends on m, so the throughput of this system depends on the scheduling discipline.

REMARK. Unfortunately we were not able to extend these results to multiple job classes. However, as we will show in the next section we were able to find exact solutions for queueing networks with BAS mechanism, more than two stations and different station types. But we have a condition on the total number of jobs allowed in the network.

3 BAS Queueing Networks with Multiple Job Classes and a Limited Number of Jobs

3.1 Model Description

Here we consider queueing networks formed by *stations*, numbered $i = 1, 2, 3, \ldots, N$. The capacity (i.e. including buffer space and the spaces in the servers) of station i is denoted b_i. We assume multiple *classes* of jobs, labeled with lowercase greek letters $(\alpha, \beta, \ldots)$. The probability that a job of class α leaving station i wants to join station j in class β is written $p_{i\alpha,j\beta}$. We will define the *relative troughputs* $e_{i\alpha}$ as any solution of the homogeneous system:

$$e_{j\beta} = \sum_{i\alpha} p_{i\alpha,j\beta} e_{i\alpha} \tag{49}$$

The name given to these quantities will be justified when we discuss performance measures for the network. The set of pairs (station, class) that a particular job may enter by the above is called a *routing chain.* We assume that the Markov chain represented by each routing chain of the network is irreducible. The routing chain that contains (i, α) will be denoted $\mathcal{R}_{i\alpha}$, while $\mathcal{R}_i$ is the set of routing chains which pass through station i. We will use r, s, ... as indices ranging over routing chains.

We will denote the total number of jobs in the buffer of station i by k_i and the total number of jobs in the buffer of station i belonging to routing chain r by k_{ir}. The total number of jobs at station i (inside the station and waiting in other stations upstream for a place in the station) is denoted by κ_i; the corresponding numbers for class α and routing chain r are $\kappa_{i\alpha}$ and κ_{ir}, respectively. The total number of jobs in routing chain r is given by K_r. The total number of jobs in the network is K.

We assume that the scheduling discipline is such that the corresponding classical network has a product form solution, i.e. there are same functions as defined before in section 2, viz, $f_i(k)$, $\phi_i(l,k)$ and $\psi_i(l,\kappa)$ with the difference in case of the latter where blocked jobs (i.e. jobs that are waiting inside other stations to enter station i) are assumed to continue the queue outside of the station, so we do not require $\psi(l,\kappa)$ to vanish for $l > b_i$. To keep jobs outside of the station separate from the jobs inside the station, we require $\psi(l,\kappa) = 0$ when $l \leq b_i$ and $\kappa > b_i$.

By the requirements we place on the scheduling discipline because of blocking, it cannot be symmetric. To get a product form solution for the general case based on considering the job waiting outside of the station as using a "shadow" space in the buffer, we must assume exponential service time distributions that are identical for all job classes in those stations that may block. Stations

whose size is effectively infinite (because they can be full without any jobs waiting outside) are exempt from this requirement.

3.2 Equivalence between Blocking and Nonblocking Networks

We set up the balance equations for the blocking network we described above under the assumption that if a station is full, there is at most one job outside that station that may try to enter it. This simplifies the description of the state, since it is then impossible for a real queue of jobs to form outside of the station waiting to enter. In escence, what happens if a job is blocked is that the station in which it is blocked becomes functionally just another space in the buffer of the blocking station. But if the blocking station had one more space in it, there would never be blocking and the network is simply a classical network. This was used by Onvural and Perros [21,22] to derive an exact solution for queueing networks with BAS and one job class. Here we extend their results to multiple job classes and give a rigourous derivation for the result.

The basic condition is that whenever station i is full, there can be at most one job outside that station in one of the routing chains that visit that station. Considering station i, this gives rise to the following:

$$\sum_{r \in \mathcal{R}_i} K_r \leq b_i + 1 \tag{50}$$

Furthermore, the job that tries to enter station i from station j has to be alone in station j, since otherwise it will affect the other jobs in it. So, all stations j that feed a station i for which relation (50) is satisfied as an equality can only be visited by the routing chains that visit station i.

We can describe the state of a station in the network under the present assumptions by the vector of the states of the stations:

$$S = (s_1, s_2, \ldots, s_N) \tag{51}$$

where the state of each individual station i is given by:

$$s_i = \begin{cases} \langle c_{i1}, c_{i2}, \ldots, c_{ik_i} \rangle & \text{nonblocking states} \\ \left\langle c_{i1}, c_{i2}, \ldots, c_{ib_i}, (j, c^*_{j1}) \right\rangle & \text{when blocking station } j \end{cases} \tag{52}$$

Here we distinguished the class of the blocked job, since it is the class the job acquires *after* finishing service and deciding to go to station i. Note also that the station from which the job comes has been recorded in the state.

We will call the states corresponding to S and s_i in (51) and (52) for the nonblocking network one gets by waiving the capacity restrictions S^+ and s_i^+, respectively. Note that many blocked states correspond to the same state of the nonblocking network.

We define the operator $T_{il\alpha,jm\beta}(S)$ applied to state S of the network as producing the state that results when the job in position l of the queue of station i (if it is of class α) is placed in position m of station j in class β. We will also use the inverse operator $T^{-1}_{il\alpha,jm\beta}$. Whenever the operation does not make sense both operators return an impossible state (a state with probability

zero). Note that in the case that station j is full under the restrictions we have imposed $T_{il\alpha,jm\beta}$ just announces that the job will enter station j in class β when a space becomes available in the blocking network. The operators can be applied the same way in the corresponding nonblocking network.

3.3 Global Balance Equations

We will set up the global balance equations for this system by considering first the effect of a job of a particular class at a given position in a station. We get the global balance equations by summing over all possible classes, positions and stations. To keep notation simple, when we refer to numbers of jobs or the class of a particular job we mean the values for state S.

The intensity at which our system enters state S due to a job in class α at position l of station i is given by:

$$\sum_{jm\beta} p_{j\beta,i\alpha}\phi_j(m,k_j+1)\mu_{j\beta}f_j(m,k_j+1)\psi_i(l,\kappa_i-1)\pi(T^{-1}_{jm\beta,il\alpha}(S)) \tag{53}$$

The intensity at which the system leaves the same state due to this job is:

$$\sum_{jm\beta} p_{i\alpha,j\beta}\phi_i(l,k_i)\mu_{i\alpha}f_i(l,k_i)\psi_j(m,\kappa_j)\pi(S) \tag{54}$$

Note that the system cannot leave a state due to a blocked job, so *job local balance* [12,16], which would equate (53) with (54) cannot possibly be satisfied. Neither does summing out over the possible job classes help, which would give *local balance* [11] We have to consider the global balance equations we get by summing out over all possible classes and positions. Summing (53) and (54) over l and α, rearranging and taking factors out of the sums whenever possible we get the global balance equations:

$$\begin{aligned}
&\sum_{il\alpha}\psi_i(l,\kappa_i-1) \\
&\qquad \sum_{jm\beta} p_{j\beta,i\alpha}\phi_j(m,k_j+1)\mu_{j\beta}f_j(m,k_j+1)\pi(T^{-1}_{jm\beta,il\alpha}(S)) \\
&= \pi(S)\sum_{il\alpha}\phi_i(l,k_i)\mu_{i\alpha}f_i(l,k_i)\sum_{jm\beta} p_{i\alpha,j\beta}\psi_j(m,\kappa_j)
\end{aligned} \tag{55}$$

We have lumped all states in which a station (with a given configuration of its queue) blocks a particular job, regardless of the station upstream in which that job is. The global balance equations that result have the same form the corresponding equations for the classical network one gets when the restrictions on station capacities are dropped. This means that both networks have the same state space (up to the lumping of blocked states) and the same state probabilities.

Actually, we have proved an equivalence. If the nonblocking network has a product form solution, so has the blocking network. The nonblocking network, given the fact pointed out when we discussed them that the scheduling discipline is *not* symmetric at blocking stations (which satisfy (50) with equality), a product form solution is given only when all jobs have the same exponential service time distribution there. Other stations underlie the normal restrictions for product form [11] since the corresponding balance equations are in no way affected. This is true even for the stations in which there could be blocked jobs.

3.4 Equilibrium Probabilities for the Blocked States

The above method only gives the probability that a job is blocked trying to enter station i, it does not give the probability that the job is blocked inside a particular uptream station. We now derive the requisite probabilities.

For a blocked state S all "neighboring" states, i.e. states from which the blocked state is entered and which are entered from the blocked state are all "classical" states without blocking. So, their probabilities are given by the expressions for the corresponding classical network. We can then set up the balance equation for entering and leaving the blocked state of interest, which gives its probability.

For the network without blocking (assuming that all service time distributions are exponential for simplicity) the probability of state S^+ is given by:

$$\pi(S^+) = \frac{1}{G}\prod_i \left[\prod_{1\le l\le k_i} \frac{1}{f_i(l)} \prod_\alpha \left(\frac{e_{i\alpha}}{\mu_{i\alpha}}\right)^{k_{i\alpha}} \right] \tag{56}$$

Here G is a normalization constant chosen so that the probabilities add up to one.

For blocked state $S = \langle c_{i1}, c_{i2}, \ldots, c_{ib_i}, (j\alpha)\rangle$ we are interested in the neighboring state in which the blocked job has not finished service yet. This state is $S_\beta = T^{-1}_{j1\beta,ib_i+1\alpha}(S)$, whose probability can be expressed in terms of state S^+ of the nonblocking network in which the blocked job is at the end of the queue of station i:

$$\pi(S_\beta) = \frac{\mu_i f_i(b_i+1)}{e_{i\alpha}} \frac{e_{j\beta}p_{j\beta,i\alpha}}{\mu_{j\beta}f_j(1)}\pi(S^+) \tag{57}$$

The rate at which state S is left is simply the rate at which jobs finish service in station i, that is $\mu_i f_i(b_i)$. So we have the balance equation around state S:

$$\mu_i f_i(b_i)\pi(S) = \sum_\beta p_{j\beta,i\alpha}\mu_{j\beta}f_j(1)\pi(S_\beta) \tag{58}$$

Using the expression (56) for the probability of state S^+, the probability of state S_β (a "classical" state, since the job is not yet blocked) can be written:

$$\pi(S_\beta) = \frac{\mu_i f_i(b_i+1)}{e_{i\alpha}} \frac{e_{j\beta}}{\mu_{j\beta}f_j(1)}\pi(S^+) \tag{59}$$

Substituting (59) into (58) we get:

$$\mu_i f_i(b_i+1)\pi(S) = \frac{\mu_i f_i(b_i)}{e_{i\alpha}}\pi(S^+)\sum_\beta e_{j\beta}p_{j\beta,i\alpha} \tag{60}$$

from which we get, since $f_i(b_i+1) = f_i(b_i)$:

$$\pi(S) = \pi(S^+)\sum_\beta e_{j\beta}p_{j\beta,i\alpha} \tag{61}$$

Note that this probability will usually depend on the class considered. Also, by the product form of $\pi(S^+)$ we can write (61) as a factor for station j, namely

$$\frac{e_{j\beta}}{\mu_j} \tag{62}$$

Usually we will be interested not in this kind of detailed state (which describes the exact order of the jobs in the station) but in the state one gets for a given vector of numbers of jobs in each class. This will multiply both $\pi(S)$ and $\pi(S^+)$ in equation (61) by the same multinomial coeficient, since the queue of station i is exactly the same for both.

By the restriction on the number of jobs of section 3.1. these are the only possibilities of a job of class α being blocked in station j by station i, except for other jobs in the rest of the network. By the product form of the solution of the network, when we sum out over all the states that have this particular configuration in stations i and j, we get a factor that is just the normalizing constant for the rest of the network, without stations i and j and without the jobs in the routing chains $\mathcal{R}_i$. This means that, in a sense, we can analyze this case in isolation.

The probability of a job in class α blocked trying to enter station i is just the probability given by (61) multiplied by the appropriate multinomial coeficient. Equation (61) considers job *classes*, so this has to be summed over all possible class memberships of the jobs inside station i. As in the classical case, the sum is nothing but the expansion of the sum of the class throughputs to the power K_r each (except for the chain which contains $i\alpha$, for which it is one less) multiplied by a constant. This allows us to write probability (61) in terms of routing chains, not classes. Using k_i to represent the vector of numbers of jobs in each routing chain in station i, we define

$$A_i(\mathrm{k}_i) = \prod_{1 \le l \le k_i} \frac{1}{f_i(l)} \prod_r \left(\frac{e_{ir}}{\mu_i}\right)^{k_{ir}} \tag{63}$$

Just the part of state S^+ that considers the last job in the queue (the blocked job) has to consider job classes. This leads to:

$$\mathcal{P}(\alpha \text{ blocked in } j \text{ by } i) = \frac{1}{G} \prod_{m \neq i,j} A_m(\mathrm{k}_m) A_i(\mathrm{k}_i) \sum_\beta \frac{p_{j\beta,i\alpha} e_{j\beta} e_{i\alpha}}{\mu_{i\alpha} f_i(b_i)} \tag{64}$$

To get the probability of a job in routing chain r blocked inside station j the above probability is summed over all classes α in chain r such that for some class β in station j we have

$$p_{j\beta,i\alpha} \neq 0 \tag{65}$$

To get the probability that a job of routing chain $\mathcal{R}$ is blocked *inside* station i is to add the above over the possible destinations of such a job.

3.5 Performance Measures

As the probabilities of the states of the blocking network are the same as in the nonblocking network, we can compute certain performance measures for the network using methods for a classical network. Note that the *numbers of jobs* in the stations are not the same as in the corresponding classical network.

3.5.1 Throughputs

The troughput of each station for a job class can be expressed as the rate at which jobs of the class under consideration finish service at the station. This is clearly the same in the blocking network

as in the corresponding network without blocking. The only difference is that in the nonblocking case the jobs move out of the station in the moment their service finishes, while a blocked job stays some additional time in the station. But it eventually moves out, before any other job in the station. So the throughputs are the same in both cases.

In the corresponding classical network, the $e_{i\alpha}$ are relative throughputs, in that:

$$\frac{e_{i\alpha}}{e_{j\beta}} = \frac{\lambda_{i\alpha}}{\lambda_{j\beta}} \tag{66}$$

for all $i\alpha$, $j\beta$. As the throughputs of the blocking network and the corresponding classical network are the same, this relation also holds in the blocking network, thus justifying the name "relative throughputs" we gave them in section 3.1.

3.5.2 Average Queue Lengths

The average queue lenths of the blocking and the corresponding classical network are different, since in the classical network the queue may extend to length $b_i + 1$, and the blocked jobs are not considered to be inside the stations in which they are blocked.

We will write $\bar{k}_{ir}$ for the mean number of jobs of routing chain r waiting for service at station i (this is the value that algorithms for the classical network give for the mean number of jobs in the station) and $\bar{q}_{ir}$ for the corresponding mean number of jobs inside the station. Then we clearly have:

$$\bar{q}_{ir} = \bar{k}_{ir} - \mathcal{P}(\text{a job in } r \text{ blocked at } i) + \mathcal{P}(\text{a job in } r \text{ blocked in } i) \tag{67}$$

The probabilities in (67) can be computed using the method outlined in section 3.1.

3.5.3 Mean Time in the Station

The time a job stays for service at a particular station is the same for both networks. But in the blocking network this includes time spent outside of the station waiting for a free place. This is not the same as the time the job stays in the station, since the job also stays in the station after its service is finished waiting for available space downstream.

The mean time a job waits for service is just the mean time in the station for the classical network. The mean time in the station is related to the mean queue length by Little's law:

$$\bar{\lambda}_{ir}\bar{t}_{ir} = \bar{q}_{ir} \tag{68}$$

from which we can easily get $\bar{t}_{ir}$, since both λ_{ir} and $\bar{q}_{ir}$ can be computed for the blocking network, λ_{ir} being just the value of the corresponding classical network and $\bar{q}_{ir}$ being obtained from (67).

4 Conclusions

Note that several variants of BAS are possible. One possibility is that the blocked job just stays in the served position in the queue of the station, denying its use to other jobs and the service effort destined to that position being lost, or the server could shut down completely when there is a blocked job in the station. We showed in section 2.2 that the distribution of the states in which no job is blocked does not depend on the number of service positions. Only the distribution of the blocked states does. So, with the above solution we can consider both alternatives by considering the case we have called FCFS (one service position) for the case where the server shuts down as soon as there is a blocked job inside the station.

The result for FCFS could also be derived using the equivalence between blocking before and after service noted by [7,19] and the method of Gordon and Newell [14]. We did not use that method here so as to show the fundamental similarity between the three cases discussed. The method used gives the complete probability distribution for this simple case, so we could determine closed form expressions for the mean number of jobs and the throughput for a symmetric case. This shows that for systems with BAS neither the mean number of jobs nor the throughput are independent of the scheduling discipline of the stations. This is one of the basic properties of the solutions for classical networks, so this reflects the greater complexity of networks with blocking.

We have shown that if whenever a station is full there is at most one job outside that station that may try to enter it, the equilibrium state probabilities of a queueing network with BAS have a simple relation to the corresponding probabilities for the same network without blocking. This condition is easy to check. Some performance measures for the blocking network can be computed directly by methods for classical networks. This is the case for throughput and the mean number of jobs waiting for service at a particular station.

The equivalence allows to compute the full equilibrium distribution for the queueing network with blocking. Using the distribution we deduce methods to compute some other performance measures (mean number of jobs and probabilities of blocking, that here are also the mean number of blocked jobs) for the blocking network based on values that are given by the classical algorithms for queueing networks. The algorithms are simple to implement using a convolution method to solve the corresponding nonblocking network.

REFERENCES

[1] I. F. Akyildiz, "Exact Product Form Solution for Queueing Networks with Blocking", *IEEE Transactions on Computers*, Vol. C-36, No. 1, January 1987, pp. 122-125.

[2] I. F. Akyildiz, "On the Exact and Approximate Throughput Analysis of Closed Queueing Networks with Blocking", *IEEE Transactions on Software Engineering*, Vol. SE-14, No. 1, January 1988, pp. 62-71.

[3] I. F. Akyildiz and H. von Brand, "Exact Solutions for Open, Closed and Mixed Queueing Networks with Rejection Blocking", *Theoretical Computer Science Journal*, No. 64, May 1989, pp. 203-219.

[4] I. F. Akyildiz and H. von Brand, "Computational Algorithms for Networks of Queues with Rejection Blocking", *Acta Informatica*, Springer Verlag, Vol. 26, pp. 559-576, July 1989.

[5] I. F. Akyildiz and H. G. Perros, "Special Issue on Queueing Networks with Finite Capacity Queues: Introduction", *Performance Evaluation*, Vol. 10, No. 3, Dec. 1989.

[6] I. F. Akyildiz and J. Liebeherr, "Optimal Deadlock Free Buffer Allocation in Multiple Chain Blocking Networks of Queues", *Proc. of the Int. Conference on the Performance of Distributed Systems and Integrated Communication Networks*, September 10-12, 1991.

[7] S. Balsamo, V. De Nitto Persone and G. Iazeoalla, "Identity and Reducibility Properties of Some Blocking and Non-blocking Mechanisms in Congested Networks", *Flow Control of Congested Networks*, NATO ASI Series, Springer Verlag, 1987, pp. 243-254.

[8] S. Balsamo and L. Donatiello, "On the Cycle Time Distribution in a Two-Stage Cyclic Network with Blocking", *IEEE Transactions on Software Engineering*, Vol. 15, No. 10, October 1989.

[9] F. Baskett, K. M. Chandy, R. R. Muntz and F. G. Palacios, "Open, Closed and Mixed Networks of Queues with Different Classes of Customers", *Journal of the ACM*, Vol. 22, Vol. 15, pp. 248-260, April 1975.

[10] P. P. Bocharov, "On the Two-Node Queueing Networks with Finite Capacity", *Proc. of the First Int. Workshop on Queueing Networks with Blocking*, North Holland, Dec. 1989, pp. 105-125.

[11] K. M. Chandy, J. H. Howard and D. F. Towsley, "Product Form and Local Balance in Queueing Networks", *Journal of the ACM*, Vol. 24, No. 2, pp. 250-263, April 1977.

[12] K. M. Chandy and A. J. Martin, "A Characterization of Product Form Queueing Networks", *Journal of the ACM*, Vol. 30, pp. 286-299, 1983.

[13] A. Conway and N. Georganas, "Queueing Networks: Exact Computational Algorithms", *MIT Press*, Fall 1989.

[14] W. J. Gordon and G. F. Newell, "Cyclic Queueing Systems with Restricted Queues", *Operations Research*, Vol. 15, 1967, pp. 266-277.

[15] F. P. Kelly, "Networks of Queues with Customers of Different Types", *Journal of Applied Probability*, Vol. 12, pp. 542-554, 1975.

[16] F. P. Kelly, "Networks of Queues", *Advances in Applied Probability*, Vol. 8, No. 2, pp. 416-432, June 1976.

[17] R. O. Onvural, "A Note on the Product Form Solutions of Multiclass Closed Queueing Networks with Blocking", *Performance Evaluation*, Vol. 10, No. 3, December 1989, pp. 247-255.

[18] R. O. Onvural, "A Survey of Closed Queueing Networks with Finite Buffers", *ACM Computing Surveys*, Vol. 22, No. 2, June 1990, pp. 83-121.

[19] R. O. Onvural and H. G. Perros, "On Equivalencies of Blocking Mechanisms in Queueing Networks with Blocking", *Operations Research Letters*, Vol. 5, Dec. 1986, pp. 293-297.

[20] R. O. Onvural and H. G. Perros, "Some Equivalencies between Closed Queueing Networks with Blocking", *Performance Evaluation Journal*, Vol. 9, Dec. 1988, pp. 111-118.

[21] R. O. Onvural and H. G. Perros, "Throughput Analysis of Cyclic Queueing Networks with Finite Buffers", *IEEE Transactions on Software Engineering*, Vol. SE-15, 1989, pp. 800-808.

[22] R. Onvural and H. G. Perros, "Equivalences between Open and Closed Queueing Networks with Finite Buffers", *Performance Evaluation*, Vol 9, 1988/1989, pp. 263-269.

[23] N. Van Dijk and H. Tijms, "Insensitivity in Two-Node Blocking Models with Applications", *Proc. Teletraffic Analysis Computer Performance Evaluation*, North Holland, pp. 329-340, 1986.

A Finite Buffer Two Class Queue with Different Scheduling and Push-Out Schemes

Xian Cheng† and Ian F. Akyildiz‡*

†AT&T Bell Labs
Room 1F-313, 480 Red Hill Road
Middletown, NJ 07748, U.S.A

‡College of Computing
Georgia Institute of Technology
Atlanta, GA 30332, U.S.A.

Abstract

We analyze an $M_1, M_2/G_1, G_2/1/N$ queue with different scheduling and push-out scheme in this paper. Our work is motivated by the study of the performance of an output link of ATM switches with two-class priority traffics. The queueing model developed in this paper is more general than that of the output link of ATM switches with two-class priority traffics. We can have general service time distributions for classes 1 and 2, and a general service discipline function, $\alpha_1(i,j)$, with $\alpha_1(i,j)$ being the probability that a class 1 packet will be served, given that there are i class 1 and j class 2 packets waiting for service. We obtain an exact solution for loss probabilities for classes 1 and 2, the queue length distribution and the mean waiting time for class 1 and an approximate calculation for the queue length distribution and mean waiting time for class 2. We show that our approximation is an upper bound and the error due to the approximation is very small when the loss probability of class 2 is small (e.g., ≤ 0.01).

1 Introduction

In this paper, we analyze an $M_1, M_2/G_1, G_2/1/N$ queue with different scheduling and push-out schemes. Our work is primarily motivated by the study of the performance of an output link of ATM switches with two-class priority traffics.

The future Broadband Integrated Services Digital Network (BISDN) will provide an integrated access that will support a wide variety of applications for its customers in a flexible and cost-effective manner. The transfer mode chosen by the CCITT for BISDN is called the ATM. ATM is a high bandwidth, low-delay, packet-like switching and multiplexing technique. ATM can switch all types of traffic, ranging from low-bit rate to high rate traffic, in a packet format of fixed length called *cell* using a simplified end-to-end protocol. Various different media such as voice, data, video and graphics can be accommodated in a ATM network. Each multimedia system requires its own grade of service (GOS). For example, voice packets are more sensitive to delay than data packets. A data packet requires a higher level of protection against loss than a voice packet. Therefore the network should be designed and controlled to satisfy these greatly differing performance requirements. Various service and buffer control mechanisms have been proposed, ranging from the dedicated buffer access for each traffic class to the shared buffer with or without push-out scheme [4], [6].

*Ian F. Akyildis was supported in part by NSF under Grant No. CCR-90-11981.

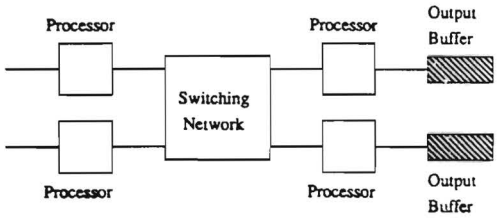


Figure 1: An ATM Switch

Consider an ATM switch consisting of a network of processors and buffers as shown in Figure 1. Cell delay and loss may occur when cells pass through the switch and the output buffer. If we assume that there are two-class traffics in the ATM switch, the output buffer can be modeled as finite buffer two-class queue as shown in Figure 2. The server in the Figure represents the trunk for the transmission of cells out of the output buffer. Our work is motivated by the study of the performance of the queueing model in Figure. 2.

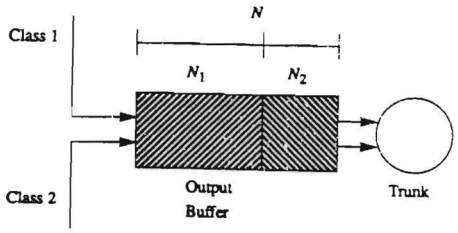


Figure 2: A Queueing Model

There is only a small number of studies on the push-out priority schemes. Doshi and Heffes [3] have described and analyzed an overload control algorithm using the push-out scheme with replacement strategy FIFO for the M/M/1/N queue. Sumita and Ozawa [4] have derived conservation laws for systems using a push-out scheme. They have also proposed a mixed head-of-line service discipline for the push-out scheme in which, when the server becomes idle, the server will serve class 1 packets first with a probability α or class 2 packets first with a $1-\alpha$. They obtain the mean waiting times for packet classes 1 and 2. Their result shows that the two mean waiting times are subject to a linear restriction. Furthermore, Hebuterne and Gravey [6] have evaluated the loss probabilities of a similar system assuming a Poisson arrival process, a deterministic service time and the replacement strategy

FIFO. Their solution is not applicable to a general service time distribution. They observe a tagged low priority packet from joining until leaving the system and derive the probabilities that this packet will either be served or discarded from the system. Kroner [7] presents a method to compute the loss probabilities of an $M_1, M_2/G/1/N$ push-out system with FIFO service discipline. He considers three different space priority mechanisms, namely, push-out scheme, partial buffer sharing, and the scheme with a separate route for each traffic class, and determines the push-out scheme as the best scheme in terms of loss probabilities. A finite-buffer priority queue $M_1, M_2/G_1, G_2/1/N$ is analyzed in [8]. However, no push-out scheme and buffer space division are considered in [8]. Most recently, Saito [10] analyzes an $MMPP1+MMPP2/G/1/K$ queue with a push-out scheme.

In this paper we present an exact method to compute loss probabilities, the distribution of the number of class 1 packets in the system and the mean waiting time of a class 1 packet. An approximate solution is given for the computation of the mean waiting time for class 2 packets. Our model in the paper differs from other analyzed push-out models in that we allow general service time distributions for classes 1 and 2, a general service discipline and a divided buffer management scheme.

This paper is organized as follows. Section 2 describes the model. Section 3 outlines the calculation of loss probabilities. In Section 4, a method for computing the steady state probabilities of the number of class 1 packets and number of class 2 packets at a service beginning time in the system is presented. Section 5 details the computation for the average number of losses of packet during a service time, which has been used in Section 3. Sections 6 and 7 derive an exact mean waiting time computation for class 1 and an approximate mean waiting time computation for class 2, respectively. Numerical examples are given in Section 8 with some discussion about the results. Finally, Section 9 concludes the paper.

2 Model Description

We consider an $M_1, M_2/G_1, G_1/1/N$ with additional features in service discipline and buffer management as will be explained shortly in this section. Since we allow general, variable length service times, we will call a customer a packet instead of a cell called in ATM networks. Two classes of packets are denoted by class 1 and class 2. The arrival process for class s ($s = 1, 2$) is Poisson with rate λ_s ($s = 1, 2$). (Note that we do not consider the bursty traffic here.) The service time of a class s ($s = 1, 2$) can be a random variable with a general probability distribution. Let $b_s(x)$ and $\overline{b}_s$ denote, respectively, the probability density function and the mean of the service time of a class s packet ($s = 1, 2$). Service times and arrival processes are independent of each other.

There are N number of total buffer spaces in the system, where N is finite and can be divided as $N = N_1 + N_2$. The number of class 1 packets waiting for service cannot be more than $N_1 - 1$. (Total number of class 1 packets in the system may be N_1 if the one currently in service is class 1.) An arrival of class 1 packet can join the system by taking an unoccupied buffer space, if it finds that there are less than $N_1 - 1$ class 1 packets waiting for service and there is an unoccupied buffer space in the system upon its arrival. An arrival of class 1 is lost if there are, upon its arrival, $N_1 - 1$ class 1 packets waiting for service in the system, even though there is an unoccupied buffer space in the system. In the contrast, an arrival of class 2 can take an unoccupied buffer space anywhere in the system upon its arrival as long as there is one. It is lost, otherwise. However, an arrival of class 1 can join the system by replacing (pushing out) a waiting class 2 packet in the system if it finds that there are less than $N_1 - 1$ class 1 packets waiting for service and that there is no unoccupied buffer space in the system upon its arrival. The class 2 packet being pushed out is lost.

The service discipline is specified by $\alpha_s(i, j)$, $s = 1, 2$, where $\alpha_1(i, j)$ is the probability that class 1 packet will be served when there are i class 1 and j class 2 packets in the system at the beginning of the service. $\alpha_2(i, j)$ can be similarly defined. Obviously, $\alpha_2(i, j) = 1 - \alpha_1(i, j)$ when $i + j > 0$. $\alpha_s(0, 0)$ ($s = 1, 2$) is undefined. We also assume that the server will not be idle as long as there is some packet in the system waiting for service. Equivalently, this is to say that $\alpha_1(i, 0) = 1$, $i > 0$, and $\alpha_2(0, j) = 1$, $j > 0$.

Using the α we can model several different scheduling disciplines in the system.

a) *Head of Line (HOL)* Scheduling

$$\alpha_1(i, j) = 1 \qquad \text{if } i > 0$$

b) *Shortest Line First (SLF)* Scheduling

$$\alpha_1(i, j) = \begin{cases} 1 & \text{if } i \le j \\ 0 & \text{if } i > j \end{cases}$$

c) *Longest Line First (LLF)* Scheduling

$$\alpha_1(i, j) = \begin{cases} 1 & \text{if } i \ge j \\ 0 & \text{if } i < j \end{cases}$$

d) *Random (RS)* Scheduling

$$\alpha_1(i, j) = p \qquad \text{if } i > 0 \ \& \ j > 0$$

i.e., the server will serve class 1 with probability p and class 2 with probability $1 - p$. We should point out that although α_1 is general, it has to be a function of the numbers of packets of two classes, and therefore, it cannot exactly model schemes that are not a function of these numbers. For example, FIFO and LIFO. Note that from the loss probabilities point of view, it does not matter in which order the packets of the same class are served and which class 2 packet will be pushed out.

3 Loss Probabilities

Packet losses occur only when the server is busy. A packet can be lost if either there is no space available in the buffer upon its arrival or it is pushed out from the buffer while waiting for

service. Let l_{12} be the loss probability for a packet of either class 1 or class 2 and s_{12} be the average number of packet losses of either class during a service time. Consider a time period T when the system reaches the steady state. On the average there are $(\lambda_1+\lambda_2)T$ arrivals from classes 1 and 2 in T, and $(1-l_{12})$ fraction of them is served. Therefore, the average number of total packet losses in T is $(\lambda_1+\lambda_2)T(1-l_{12})s_{12}$. By definition, l_{12} is the ratio of the average number of total losses in T to the average number of total arrivals in T. Thus,

$$l_{12} = \frac{(\lambda_1+\lambda_2)T(1-l_{12})s_{12}}{(\lambda_1+\lambda_2)T} = (1-l_{12})s_{12} \qquad (1)$$

From (1) we get

$$l_{12} = \frac{s_{12}}{1+s_{12}} \qquad (2)$$

The computation of s_{12} will be shown later (Section 5).

Similarly, let l_1 be the loss probability of a class 1 packet and s_1 be the average number of losses of class 1 packets during a service time. We have

$$l_1 = \frac{(\lambda_1+\lambda_2)T(1-l_{12})s_1}{\lambda_1 T} = \frac{(\lambda_1+\lambda_2)(1-l_{12})s_1}{\lambda_1} \qquad (3)$$

Finally, let l_2 be the loss probability of a class 2 packet. Using

$$(\lambda_1+\lambda_2)l_{12} = \lambda_1 l_1 + \lambda_2 l_2 \qquad (4)$$

we obtain

$$l_2 = \frac{(\lambda_1+\lambda_2)l_{12} - \lambda_1 l_1}{\lambda_2} \qquad (5)$$

s_1 will be computed in Section 5.

4 Steady State Probabilities

The average number of packet losses during a service time can be computed by conditioning on the number of class 1 and the number of 2 packets in the system at the beginning of the service time. In this section, we will compute the probabilistic distribution of the numbers of class 1 and class 2 packets in the system at the beginning of a service time. We proceed as follows. First, the distribution of the numbers of packets left in the system at a packet's departure time is computed, and then the distribution of the numbers of packets at the beginning of a service time is derived from the departure time distribution.

Let (i,j) denote that there are i class 1 packets and j class 2 packets in the queue at a packet's departure time. Since we restrict our view at a packet's departure time, (i,j) constitutes a Markov chain (imbedded Markov chain), where $0 \le i < N_1$, $j \ge 0$ and $i+j \le N-1$. Let $p(i,j)$, $0 \le i < N_1$, $j \ge 0$ and $i+j \le N-1$, be the steady state probability that the system is in state (i,j) at a packet's departure time and $P_{(i,j);(k,l)}$ be the one step transition probability from state (i,j) to state (k,l). Clearly, $P_{(i,j);(k,l)}$ is a function of arrival rates and service time. The Markov chain is totally determined by $P_{(i,j);(k,l)}$. To facilitate the expression for $P_{(i,j);(k,l)}$, the following definitions are introduced.

Definition 1. $I(n,\lambda,b(x))$ is the probability that there are exactly n Poisson arrivals with arrival rate λ during a service time of which the probability density function (pdf) is $b(x)$.

$$\begin{aligned} I(n,\lambda,b(x)) &= \int_0^\infty \frac{(\lambda x)^n}{n!} e^{-\lambda x} b(x)dx \\ &= \frac{\lambda^n}{n!}\int_0^\infty x^n e^{-\lambda x} b(x)dx \end{aligned} \qquad (6)$$

Definition 2. $I(\ge n,\lambda,b(x))$ is the probability that there are at least n Poisson arrivals with rate λ during a service time of which the pdf is $b(x)$.

$$I(\ge n,\lambda,b(x)) = 1 - \sum_{i=0}^{n-1} I(i,\lambda,b(x)) \qquad (7)$$

Definition 3. $II(n_1,n_2,\lambda_1,\lambda_2,b(x))$ is the probability that there are exactly n_1 and n_2 arrivals from Poisson processes with arrival rates λ_1 and λ_2, respectively, during a service time of which the pdf is $b(x)$.

$$\begin{aligned} &II(n_1,n_2,\lambda_1,\lambda_2,b(x)) \\ &= \int_0^\infty \frac{(\lambda_1 x)^{n_1}}{n_1!} e^{-\lambda_1 x} \frac{(\lambda_2 x)^{n_2}}{n_2!} e^{-\lambda_2 x} b(x)dx \\ &= \frac{{\lambda_1}^{n_1}{\lambda_2}^{n_2}(n_1+n_2)!}{(\lambda_1+\lambda_2)^{n_1+n_2} n_1! n_2!} I(n_1+n_2,\lambda_1+\lambda_2,b(x)) \end{aligned} \qquad (8)$$

Definition 4. $II(\ge n_1,n_2,\lambda_1,\lambda_2,b(x))$ is the probability that there are at least n_1 and exactly n_2 arrivals from Poisson processes with arrival rates λ_1 and λ_2, respectively, during a service time of which the pdf is $b(x)$.

$$\begin{aligned} &II(\ge n_1,n_2,\lambda_1,\lambda_2,b(x)) \\ &= I(n_2,\lambda_2,b(x)) - \sum_{i=0}^{n_1-1} II(i,n_2,\lambda_1,\lambda_2,b(x)) \end{aligned} \qquad (9)$$

The following two probabilities are similarly defined. They are

$$\begin{aligned} &II(n_1,\ge n_2,\lambda_1,\lambda_2,b(x)) \\ &\quad = II(\ge n_2,n_1,\lambda_2,\lambda_1,b(x)) \end{aligned} \qquad (10)$$

and

$$\begin{aligned} &II(\ge n_1,\ge n_2,\lambda_1,\lambda_2,b(x)) \\ &= \int_0^\infty \sum_{i=n_1}^{\infty}\sum_{j=n_2}^{\infty} \frac{(\lambda_1 x)^i}{i!} e^{-\lambda_1 x} \frac{(\lambda_2 x)^j}{j!} e^{-\lambda_2 x} b(x)dx \\ &= I(\ge n_2,\lambda_2,b(x)) - \sum_{i=0}^{n_1-1} II(i,\ge n_2,\lambda_1,\lambda_2,b(x)) \end{aligned} \qquad (11)$$

If $n_2 < 0$ in the above definitions, the computation should not count the arrival process related to n_2. For example, $II(n_1,n_2,\lambda_1,\lambda_2,b(x)) = I(n_1,\lambda_1,b(x))$ if $n_2 < 0$.

Now we are ready to compute $P_{(i,j);(k,l)}$.

Case 1. For $i=0$ and $j=0$: the class of the packet that will be served next depends on the class from which the next packet comes. Since both arrival processes are Poisson, with probability $\frac{\lambda_1}{\lambda_1+\lambda_2}$ the next packet comes from class 1 and with probability $\frac{\lambda_2}{\lambda_1+\lambda_2}$ from class 2. Therefore, we have:

$$P_{(0,0);(k,l)} = \begin{cases} [\frac{\lambda_1}{\lambda_1+\lambda_2} II(k,l,\lambda_1,\lambda_2,b_1(x))+ \\ \frac{\lambda_2}{\lambda_1+\lambda_2} II(k,l,\lambda_1,\lambda_2,b_2(x))] \\ \quad if\ k < N_1 - 1\ \&\ k+l < N-1 \\ [\frac{\lambda_1}{\lambda_1+\lambda_2} II(k,\geq l,\lambda_1,\lambda_2,b_1(x))+ \\ \frac{\lambda_2}{\lambda_1+\lambda_2} II(k,\geq l,\lambda_1,\lambda_2,b_2(x))] \\ \quad if\ k < N_1 - 1\ \&\ k+l = N-1 \\ [\frac{\lambda_1}{\lambda_1+\lambda_2} II(\geq k,l,\lambda_1,\lambda_2,b_1(x))+ \\ \frac{\lambda_2}{\lambda_1+\lambda_2} II(\geq k,l,\lambda_1,\lambda_2,b_2(x))] \\ \quad if\ k = N_1 - 1\ \&\ k+l < N-1 \\ [\frac{\lambda_1}{\lambda_1+\lambda_2} II(\geq k,\geq l,\lambda_1,\lambda_2,b_1(x))+ \\ \frac{\lambda_2}{\lambda_1+\lambda_2} II(\geq k,\geq l,\lambda_1,\lambda_2,b_2(x))] \\ \quad if\ k = N_1 - 1\ \&\ k+l = N-1 \end{cases} \quad (12)$$

Case 2. For $i = 0$ and $j > 0$: a packet of class 2 will be served. Let $\Delta_1 = k$ and $\Delta_2 = l-(j-1)$. Δ_1 and Δ_2 indicate, respectively, numbers of changes of classes 1 and 2 packets during a service time. Δ_2 may become negative if $j-1 > N_2$ and $\Delta_1 > N-j$. Therefore we have:

$$P_{(0,j);(k,l)} = \begin{cases} 0 \\ \quad if\ \Delta_2 < 0\ \&\ k+l < N-1 \\ II(\Delta_1,\Delta_2,\lambda_1,\lambda_2,b_2(x)) \\ \quad if\ k < N_1 - 1\ \&\ k+l < N-1 \\ II(\Delta_1,\geq \Delta_2,\lambda_1,\lambda_2,b_2(x)) \\ \quad if\ k < N_1 - 1\ \&\ k+l = N-1 \\ II(\geq \Delta_1,\Delta_2,\lambda_1,\lambda_2,b_2(x)) \\ \quad if\ k = N_1 - 1\ \&\ k+l < N-1 \\ II(\geq \Delta_1,\geq \Delta_2,\lambda_1,\lambda_2,b_2(x)) \\ \quad if\ k = N_1 - 1\ \&\ k+l = N-1 \end{cases} \quad (13)$$

Case 3. For $i > 0$ and $j = 0$: a packet of class 1 will be served. Let $\Delta_1 = k-(i-1)$ and $\Delta_2 = l$. Δ_1 and Δ_2 have the same interpretation as that in case 2. Note that Δ_1 cannot be negative. We have

$$P_{(i,0);(k,l)} = \begin{cases} 0 \\ \quad if\ \Delta_1 < 0 \\ II(\Delta_1,\Delta_2,\lambda_1,\lambda_2,b_1(x)) \\ \quad if\ k < N_1 - 1\ \&\ k+l < N-1 \\ II(\Delta_1,\geq \Delta_2,\lambda_1,\lambda_2,b_1(x)) \\ \quad if\ k < N_1 - 1\ \&\ k+l = N-1 \\ II(\geq \Delta_1,\Delta_2,\lambda_1,\lambda_2,b_1(x)) \\ \quad if\ k = N_1 - 1\ \&\ k+l < N-1 \\ II(\geq \Delta_1,\geq \Delta_2,\lambda_1,\lambda_2,b_1(x)) \\ \quad if\ k = N_1 - 1\ \&\ k+l = N-1 \end{cases} \quad (14)$$

Case 4. For $i > 0$ and $j > 0$: a packet of class s ($s = 1, 2$) will be served with probability $\alpha_s(i,j)$. Let $\Delta_1^1 = k-(i-1)$, $\Delta_2^1 = l-j$, $\Delta_1^2 = k-i$, and $\Delta_2^2 = l-(j-1)$. Δ_1^1 and Δ_2^1 indicate, respectively, the numbers of changes of classes 1 and 2 packets in a class 1 packet service time, and Δ_1^2 and Δ_2^2 have the similar interpretation except that in a class 2 packet service time. Note that it is impossible to have $\Delta_1^1 < 0$ or $\Delta_2^1 < 0$ & $k+l < N-1$ when a class 1 packet is being served and $\Delta_1^2 < 0$ or $\Delta_2^2 < 0$ & $k+l < N-1$ when a class 2 packet is being served. Let

$$\delta(\Delta_1,\Delta_2) = \begin{cases} 0 \\ \quad if\ \Delta_1 < 0 \\ \quad or\ \Delta_2 < 0\ \&\ k+l < N-1 \\ 1 \\ \quad otherwise \end{cases} \quad (15)$$

Thus

$$P_{(i,j);(k,l)} = \begin{cases} \alpha_1(i,j)\delta(\Delta_1^1,\Delta_2^1) II(\Delta_1^1,\Delta_2^1,\lambda_1,\lambda_2,b_1(x))+ \\ \alpha_2(i,j)\delta(\Delta_1^2,\Delta_2^2) II(\Delta_1^2,\Delta_2^2,\lambda_1,\lambda_2,b_2(x)) \\ \quad if\ k < N_1 - 1\ \&\ k+l < N-1 \\ \alpha_1(i,j)\delta(\Delta_1^1,\Delta_2^1) II(\Delta_1^1,\geq \Delta_2^1,\lambda_1,\lambda_2,b_1(x))+ \\ \alpha_2(i,j)\delta(\Delta_1^2,\Delta_2^2) II(\Delta_1^2,\geq \Delta_2^2,\lambda_1,\lambda_2,b_2(x)) \\ \quad if\ k < N_1 - 1\ \&\ k+l = N-1 \\ \alpha_1(i,j)\delta(\Delta_1^1,\Delta_2^1) II(\geq \Delta_1^1,\Delta_2^1,\lambda_1,\lambda_2,b_1(x))+ \\ \alpha_2(i,j)\delta(\Delta_1^2,\Delta_2^2) II(\geq \Delta_1^2,\Delta_2^2,\lambda_1,\lambda_2,b_2(x)) \\ \quad if\ k = N_1 - 1\ \&\ k+l < N-1 \\ \alpha_1(i,j)\delta(\Delta_1^1,\Delta_2^1) II(\geq \Delta_1^1,\geq \Delta_2^1,\lambda_1,\lambda_2,b_1(x))+ \\ \alpha_2(i,j)\delta(\Delta_1^2,\Delta_2^2) II(\geq \Delta_1^2,\geq \Delta_2^2,\lambda_1,\lambda_2,b_2(x)) \\ \quad if\ k = N_1 - 1\ \&\ k+l = N-1 \end{cases} \quad (16)$$

This completes the computation for $P_{(i,j);(k,l)}$, where $0 \leq i,k < N_1$; $j,l \geq 0$ and $i+j, k+l < N$.

The steady state probabilities, $p(i,j)$, should observe the law of conservation:

$$p(i,j) = \sum_{all\ (k,l)} p(k,l) P_{(k,l),(i,j)} \quad for\ 0 \leq i \leq N_1;\ j \geq 0;\ i+j \leq N-1 \quad (17)$$

and also

$$\sum_{all\ (i,j)} p(i,j) = 1 \quad (18)$$

We can compute the values of $p(i,j)$ by solving equations (17) and (18) numerically, which involves $\frac{N_1(2N-N1+1)}{2}$ independent linear equations.

Let q(i, j) be the probability that there are i class 1 packets and j class 2 packets at the beginning of a service time. Except for the first packet, the beginning of a service is preceded by the departure of the last packet served. There is then a one-to-one correspondence between a packet's departure and the beginning of the service of the next packet. If there is some packet left in the system at a packet's departure time, then the beginning of the service time for the next packet coincides with the departure time and they should ob-

serve the same packets left in the system. However, if there is no packet left at a departure time, there will be either $(1,0)$ or $(0,1)$ packet in the system at the beginning of the next service time depending on from which class the next packet comes. It is clear now that the possible state (i,j) at the beginning of a service is $i+j \geq 1$ and $i+j \leq N-1$ for $N \geq 2$. So $q(i,j)$ is computed as follows.

1. For $N=1$, the only possible states are $(1,0)$ and $(0,1)$:

$$q(1,0) = \frac{\lambda_1}{\lambda_1+\lambda_2}$$

and

$$q(0,1) = \frac{\lambda_2}{\lambda_1+\lambda_2}$$

2. For $N \geq 2$:

$$q(i,j) = \begin{cases} \frac{\lambda_1}{\lambda_1+\lambda_2}p(0,0) \\ \quad if\ i=1\ \&\ j=0\ \&\ N_1=1 \\ p(1,0)+\frac{\lambda_1}{\lambda_1+\lambda_2}p(0,0) \\ \quad if\ i=1\ \&\ j=0\ \&\ N_1>1 \\ p(0,1)+\frac{\lambda_2}{\lambda_1+\lambda_2}p(0,0) \\ \quad if\ i=0\ \&\ j=1 \\ p(i,j) \\ \quad if\ i+j\geq 2\ \&\ i+j\leq N-1 \end{cases} \quad (19)$$

5 Average Number of Losses during a Service Time

We define $L(n,\lambda,b(x))$ as the average number of arrivals after the first n arrivals of a Poisson arrival process with rate λ during a service time whose pdf is $b(x)$, i.e., $L(n,\lambda,b(x))$ is the average number of arrivals counted after the first n arrivals during a service time. By definition

$$\begin{aligned} & L(n,\lambda,b(x)) \\ &= \sum_{k=n+1}^{\infty}\int_0^{\infty}(k-n)\frac{(\lambda x)^k}{k!}e^{-\lambda x}b(x)dx \\ &= \lambda\bar{b} - \sum_{k=1}^{n} kI(k,\lambda,b(x)) - nI(\geq(n+1),\lambda,b(x)) \end{aligned} \quad (20)$$

where $\bar{b}$ is the mean of $b(x)$. Suppose there are $(i,j), i>0$ and $j>0$, packets in the system (including the one which is going to receive service) at the beginning of a service time. Consider the number of class 1 packets that may be lost during the service time. If a class 1 packet is served, then the first (N_1-i) arrivals of class 1 packets during the service time can join the system. After that, all arrivals are lost due to the fact that there are (N_1-1) class 1 packets waiting in the queue. Therefore, the average number of class 1 packets lost during a service time which begins with i class 1 packets and j class 2 packets is equal to the average number of class 1 packets arrived after the first (N_1-i) class 1 arrivals if the packet in service is class 1 or is equal to the average number of class 1 packets arrived after the first $(N_1-(i+1))$ class 1 arrivals if the packet in service is class 2. Therefore,

$$\begin{aligned} s_1 &= \sum_{i>0,j\geq 0} q(i,j)\alpha_1(i,j)L(N_1-i,\lambda_1,b_1(x)) \\ &+ \sum_{j>0,i\geq 0} q(i,j)\alpha_2(i,j)L(N_1-i-1,\lambda_1,b_2(x)) \end{aligned} \quad (21)$$

The idea for computing s_{12}, the average number of losses of packets of the two classes during a service time is similar but more complicated. Again, suppose there are (i,j) packets in the system at the beginning of a service time. First, let us consider the case where the next packet to be served is a class 1 packet. Let $t = Min\{N_1-i, N-i-j\}$ and $y = Max\{0, N_2-j\}$. t can be thought as the maximum number of class 1 arrivals during the service time which result in no packets being lost or pushed out, and y is the number of unoccupied buffer spaces that only class 2 packets can take. $(N-i-j)$ is the total number of unoccupied buffer spaces at the beginning of the service time. Assuming that there are k and l arrivals from classes 1 and 2, respectively, during a service time beginning at state (i,j), the number of total losses of packets of the two classes during the service time is

$$s_{12}\,|_{(i,j);(k,l)} = \begin{cases} l-(N-i-j-k) \\ \quad if\ k\leq t\ \&\ l>N-i-j-k \\ k-t \\ \quad if\ k>t\ \&\ l\leq y \\ k-t+l-y \\ \quad if\ k>t\ \&\ l>y \\ 0 \\ \quad otherwise \end{cases} \quad (22)$$

Therefore, the average number of packets of the two classes lost during a class 1 service time beginning at (i,j) is

$$\begin{aligned} & s_{12}\,|_{(i,j)} \\ &= \int_0^{\infty}\sum_{k=0}^{t}\sum_{l=N-i-j-k+1}^{\infty}\frac{(\lambda_1 x)^k}{k!}e^{-\lambda_1 x}\frac{(\lambda_2 x)^l}{l!}e^{-\lambda_2 x} \\ & (l-(N-i-j-k))b_1(x)dx + \int_0^{\infty}\sum_{k=t+1}^{\infty}\sum_{l=0}^{y}\frac{(\lambda_1 x)^k}{k!}e^{-\lambda_1 x} \\ & \frac{(\lambda_2 x)^l}{l!}e^{-\lambda_2 x}(k-t)b_1(x)dx + \int_0^{\infty}\sum_{k=t+1}^{\infty}\sum_{l=y+1}^{\infty}\frac{(\lambda_1 x)^k}{k!}e^{-\lambda_1 x} \\ & \frac{(\lambda_2 x)^l}{l!}e^{-\lambda_2 x}(k-t+l-y)b_1(x)dx \\ &= \frac{\lambda_2}{\lambda_1}\sum_{k=0}^{t}(k+1)II(k+1,\geq(N-i-j-k),\lambda_1,\lambda_2,b_1(x)) - \\ & \sum_{k=0}^{t}(N-i-j-k)II(k,\geq(N-i-j-k+1),\lambda_1,\lambda_2,b_1(x)) + \\ & (\lambda_1+\lambda_2)\bar{b}_1 - \sum_{k=0}^{t}kI(k,\lambda_1,b_1(x)) - tI(\geq(t+1),\lambda_1,b_1(x)) - \\ & \sum_{l=0}^{y}lI(l,\lambda_2,b_1(x)) - \frac{\lambda_2}{\lambda_1}\sum_{k=0}^{t}(k+1)II((k+1),\geq y,\lambda_1,\lambda_2,b_1(x)) - \\ & yII(\geq(t+1),\geq(y+1),\lambda_1,\lambda_2,b_1(x)) \end{aligned} \quad (23)$$

The case of the next packet to be served being a class 2 packet can be derived similarly. The result will be the same except that t and y are calculated slightly differently:

$t = Min\{N_1-(i+1), N-i-j\}$ and $y = Max\{0, N_2-(j-1)\}$, where $j \geq 1$.

Let

$$
\begin{aligned}
&g(i,j,t,y,b(x)) = \\
&\frac{\lambda_2}{\lambda_1}\sum_{k=0}^{t}(k+1)II(k+1,\geq(N-i-j-k),\lambda_1,\lambda_2,b(x)) - \\
&\sum_{k=0}^{t}(N-i-j-k)II(k,\geq(N-i-j-k+1),\lambda_1,\lambda_2,b(x)) + \\
&(\lambda_1+\lambda_2)\overline{b} - \sum_{k=0}^{t}kI(k,\lambda_1,b(x)) - tI(\geq(t+1),\lambda_1,b(x)) - \\
&\sum_{l=0}^{y}lI(l,\lambda_2,b(x)) - \frac{\lambda_2}{\lambda_1}\sum_{k=0}^{t}(k+1)II((k+1),\geq y,\lambda_1,\lambda_2,b(x)) - \\
&yII(\geq(t+1),\geq(y+1),\lambda_1,\lambda_2,b(x)) \qquad (24)
\end{aligned}
$$

where $\overline{b} = \int_0^\infty xb(x)dx$. Then, s_{12}, the mean number of losses of packets of the two classes in a service time, is

$$
\begin{aligned}
s_{12} &= \sum_{i>0,j\geq 0} q(i,j)\alpha_1(i,j)g(i,j,t_1(i,j),y_1(j),b_1(x)) \\
&+ \sum_{j>0,i\geq 0} q(i,j)\alpha_2(i,j)g(i,j,t_2(i,j),y_2(j),b_2(x)) \qquad (25)
\end{aligned}
$$

where

$$
\begin{aligned}
t_1(i,j) &= Min\{N_1-i, N-i-j\} \\
g_1(j) &= Max\{0, N_2-j\} \\
t_2(i,j) &= Min\{N_1-(i+1), N-i-j\} \\
g_2(j) &= Max\{0, N_2-(j-1)\}
\end{aligned} \qquad (26)
$$

Special Case: if $N_1 = N$, i.e., class 1 packets can take any buffer space in the system, the computation for s_{12} is much simpler. Suppose there are i and j classes 1 and 2 packets, respectively, at the beginning of a service time, any arrival after the first $(N-i-j)$ arrivals from both classes either is lost or pushes out a class 2 packet. Therefore, (25) is simplified to the following form:

$$
\begin{aligned}
s_{12} &= \sum_{i>0,j\geq 0} q(i,j)\alpha_1(i,j)L(N-i-j,\lambda_1+\lambda_2,b_1(x)) \\
&+ \sum_{j>0,i\geq 0} q(i,j)\alpha_2(i,j)L(N-i-j,\lambda_1+\lambda_2,b_2(x)) \qquad (27)
\end{aligned}
$$

6 Exact Computation of the Queue Length Distribution and the Mean Waiting Time for Class 1

In this section, we compute the probability of i, $0 \leq i \leq N_1$, class 1 packets in the system at a random time. The result is then used to compute the mean waiting time of a class 1 packet. Since Poisson arrivals see time average [9], the probability that there are i, $0 \leq i \leq N_1$, class 1 packets in the system at a random time is equal to the probability that there are i class 1 packets in the system at the arrival time of a class 1 packet. So we will compute the probability from the point of view of an arriving class 1 packet. As before, T is used to denote a period of time when the system is in the steady state.

Let a_{idle} be the probability that a class 1 packet finds the server idle upon its arrival. This is possible only when there is no packet in the system at a service completion time and the next arrival is a class 1 packet. Therefore, we have

$$
\begin{aligned}
a_{idle} &= \frac{(\lambda_1+\lambda_2)T(1-l_{12})p(0,0)\frac{\lambda_1}{\lambda_1+\lambda_2}}{\lambda_1 T} \\
&= (1-l_{12})p(0,0) \qquad (28)
\end{aligned}
$$

Let a_k, $0 \leq k \leq N_1$, be the average number of class 1 arrivals which see k class 1 packets in the system upon their arrivals during a service time. The population of class 1 packets can be divided into two sets: those lost upon their arrivals and those served. The class 1 packets lost can see only N_1-1 or N_1 class 1 packets in the system upon their arrivals, while class 1 packets served can see $0 \leq k \leq N_1-1$ class 1 packets in the system upon their arrivals. The number of losses of class 1 packet in a service time is

$$
s_1 = a'_{N_1} + a'_{N_1-1} \qquad (29)
$$

where

$$
a'_{N_1} = \sum_{i>0,j\geq 0} q(i,j)\alpha_1(i,j)L(N_1-i,\lambda_1,b_1(x)) \qquad (30)
$$

and

$$
a'_{N_1-1} = \sum_{j>0,i\geq 0} q(i,j)\alpha_2(i,j)L(N_1-i-1,\lambda_1,b_2(x)) \qquad (31)
$$

a'_{N_1-1} and a'_{N_1} are the average numbers of class 1 packets lost during a service time which see N_1-1 and N_1 class 1 packets in the system upon their arrivals, respectively.

The average number of class 1 packets served which see k, $0 \leq k \leq N_1-1$, class 1 packets in the system upon their arrivals during a service time can be computed as follows. If there is no class 1 packet at the beginning of a service, there should be at least $k+1$ class 1 arrivals during the service time for only the $(k+1)st$ arrival will observe k class 1 packets in the system upon its arrival. If there are i, $i \geq 1$, class 1 packets in the system at the beginning of a service, there should be at least $k-i+1$ class 1 arrivals during the service time for only the $(k-i+1)st$ arrival will observe k class 1 packets in the system upon its arrival. Obviously, $i \leq k$ and $i+1 < N_1$ if a packet of class 2 is in service. Therefore we have

$$
a_k = \begin{cases}
\sum_{i\leq k} q(i,j)\alpha_1(i,j)I(\geq(k-i+1),\lambda_1,b_1(x)) + \\
\sum_{i\leq k} q(i,j)\alpha_2(i,j)I(\geq(k-i+1),\lambda_1,b_2(x)) \\
\qquad\qquad if\ 0 \leq k < N_1-1 \\
\sum_{i\leq k} q(i,j)\alpha_1(i,j)I(\geq(k-i+1),\lambda_1,b_1(x)) + a'_{N_1-1} \\
\qquad\qquad if\ k = N_1-1 \\
a'_{N_1} \\
\qquad\qquad if\ k = N_1
\end{cases} \qquad (32)
$$

Now let B_k, $0 \leq k \leq N_1$, denote the probability that a class 1 packet finds that there are k class 1 packets in the system upon its arrival. B_k is then

a) If $k = 0$:

$$B_0 = \frac{(\lambda_1+\lambda_2)T(1-l_{12})a_0}{\lambda_1 T} + a_{idle} = \frac{(\lambda_1+\lambda_2)(1-l_{12})a_0}{\lambda_1} + a_{idle} \quad (33)$$

b) If $k = 1, 2, ..., N_1$:

$$B_k = \frac{(\lambda_1+\lambda_2)T(1-l_{12})a_k}{\lambda_1 T} = \frac{(\lambda_1+\lambda_2)(1-l_{12})a_k}{\lambda_1} \quad (34)$$

It can be verified that

$$\sum_{k=0}^{N_1} B_k = 1 \quad (35)$$

The mean system time of a class 1 packet can be computed by Little's law. The average number of packets in the system equals to the mean system time multiplied by the effective arrival rate. Therefore, the mean waiting time of a class 1 packet, $\overline{w}_1$, is

$$\overline{w}_1 = \frac{\sum_{k=1}^{N_1} kB_k}{(1-l_1)\lambda_1} - \overline{b}_1 \quad (36)$$

7 Approximate Mean Waiting Time for Class 2

We are unable to compute the mean waiting time of a class 2 packet exactly due to the fact that a class 2 packet may get pushed out after joining the waiting queue. However, in the context of an ATM switch, the loss probability of a class 2 packet, which can be computed exactly by the method described in section 3, is usually very small. When the loss probability is small, we can analyze approximately the mean waiting time of a class 2 packet by overlooking part of lost packets. Particularly, in the following we will present an approximate method of computing the mean waiting time of a class 2 packet with the assumption that only those arrivals of class 2 packets which arrive after the first $(N-i-j)$ arrivals of class 2 packets during a service time beginning with i class 1 and j class 2 packets in the system will be lost and there is no push-out loss.

If there are i class 1 and j class packets in the system at the beginning of service, $(N-i-j)$ is the number of unoccupied buffer spaces at the beginning of the service. An arrival among the first $N-i-j$ arrivals of class 2 packets during a service may or may not be lost, depending on the number of class 1 packets arrived ahead of it during the service time. The arrivals of class 2 packets after the first $N-i-j$ arrivals of class 2 are always lost. Thus the number of actual losses of class 2 packets in computing the mean waiting time of a class 2 packet is reduced. We will comment on the accuracy of it shortly. As before, we compute the mean number of class 2 packets in the system at a random time first, which can be carried out equivalently by computing the mean number seen by an arriving class 2 packet. We then use Little's law to compute the mean waiting time. Because of the assumption, the mean number of class 2 packets in the system at a random time computed this way is greater than the actual mean number of class 2 packets served. So the mean waiting time computed with the assumption is an upper bound of the actual mean waiting time. We can also estimate a lower bound of the mean waiting time of a class 2 packet as follows. Let N_u be the upper bound of the mean number of class 2 packets computed with the assumption. As the assumption suggests, the actual mean number of class 2 packets in the system is greater than $N_u(1-l_2)$. So the error in the mean waiting time introduced by the assumption is, by Little's law, no more than l_2 fraction of the actual mean waiting time. For example, suppose the loss probability of a class 2 packet is 10^{-2}, our approximate computation of the mean waiting time of a class 2 packet will have an error of less than 1 percent of that of exact computation, which is probably acceptable for practical interest.

Similar to the previous section, let c_{idle} be the probability that a class 2 packet finds the server idle upon its arrival, then

$$c_{idle} = (1-l_{12})p(0,0) \quad (37)$$

Let c_k, $0 \le k \le N$, be the average number of class 2 arrivals which see k class 2 packets in the system upon their arrivals during a service time. The arrivals of class 2 during a service time can be divided into two sets, depending on whether they are lost or not upon their arrivals. Let c_k', $N_2+1 \le k \le N$, be the average number of class 2 packets in a service time which see k class 2 packets in the system upon their arrivals and are lost at the same time due to no unoccupied buffer space, then

$$c_k' = \sum_{j<k} q(N-k,j)\alpha_1(N-k,j)L(k-j,\lambda_2,b_1(x)) + \sum_{j<k} q(N-k,j)\alpha_2(N-k,j)L(k-j,\lambda_2,b_2(x)) \quad for\ N_2+1 \le k \le N \quad (38)$$

and c_k, $0 \le k \le N$ is

$$c_k = \begin{cases} [\sum_{j\le k\ \&\ i+k<N} q(i,j)\alpha_1(i,j)I(\ge(k-j+1),\lambda_2,b_1(x)) + \sum_{j\le k\ \&\ i+k<N} q(i,j)\alpha_2(i,j)I(\ge(k-j+1),\lambda_2,b_2(x))] & if\ k < N_2 \\ [\sum_{j\le k\ \&\ i+k<N} q(i,j)\alpha_1(i,j)I(\ge(k-j+1),\lambda_2,b_1(x)) + \sum_{j\le k\ \&\ i+k<N} q(i,j)\alpha_2(i,j)I(\ge(k-j+1),\lambda_2,b_2(x)) + c_k'] & if\ N_2 \le k < N \\ c_N' & if\ k = N \end{cases} \quad (39)$$

Let D_k, $0 \le k \le N$, denote the probability that a class 2 packet finds there are k class 2 packets in the system upon its arrival. Then

1. If $k = 0$:

$$D_0 = \frac{(\lambda_1+\lambda_2)T(1-l_{12})c_0}{\lambda_2 T} + a_{idle} = \frac{(\lambda_1+\lambda_2)(1-l_{12})c_0}{\lambda_2} + a_{idle} \quad (40)$$

2. If $0 < k \le N$:

$$D_k = \frac{(\lambda_1+\lambda_2)T(1-l_{12})c_k}{\lambda_2 T} = \frac{(\lambda_1+\lambda_2)(1-l_{12})c_k}{\lambda_2} \quad (41)$$

The mean waiting time of a class 2 packet, $\overline{w}_2$, is approximately

$$\overline{w}_2 \approx \frac{\sum_{k=1}^{N} kD_k}{(1-l_2)\lambda_2} - \overline{b}_2 \quad (42)$$

8 Numerical Examples

In this section we present some of the experimental computations conducted in the study. It is assumed, in all of our examples, that service times for two classes are constant and equal to 1. Three service disciplines, namely HOL, SLF and LLF, are used for comparison. Let $\rho = \lambda_1 + \lambda_2$ be the total load to the system (since the service time is normalized to 1). An admissible load with respect to a certain GOS for classes 1 and 2, which is specified in terms of loss probabilities and mean waiting times for classes 1 and 2 in the study, is the maximum total load without violating the GOS. At a given load, three different mixes of loads from classes 1 and 2 are tried. The three mixes are $\lambda_1 = \lambda_2$, $\lambda_1 = 2\lambda_2$ and $2\lambda_1 = \lambda_2$.

The first set of examples (Figures 3, 4 and 5), displays relationship between admissible loads and total buffer sizes. The same GOS are used in three figures with $\lambda_1 = \lambda_2$ in Figure 3, $\lambda_1 = 2\lambda_2$ in Figure 4 and $2\lambda_1 = \lambda_2$ in Figure 5. The GOS is

$$l_1 \le 10^{-10}, \quad l_2 \le 10^{-6}$$

$$\overline{w}_1 \le 1.5, \quad \overline{w}_2 \le 5$$

The admissible loads are represented on y-axes and total buffer sizes N are on x-axes where $N = N_1$ is assumed. Three curves in each figure correspond to three service disciplines.

As we can see, HOL administers the largest admissible loads with respect to the GOS used here. This is true not only for different buffer size but also for different load mixes. In Figure 3, the limiting factor of admissible load is loss probability of class 2 in all three service disciplines. In Figure 4, where $\lambda_1 = 2\lambda_2$, the limiting factor differs with service disciplines. For HOL, the limiting factor is the loss probability of class 1 when N, the total buffer size, is less than or equal to 20 and the loss probability of class 2 when $N > 20$. However, at $N = 40$ both the loss probability and the mean waiting time of class 2 approach the GOS limit simultaneously. For SLF, the limiting factor is the loss probability of class 1 when $N \le 38$ and the mean waiting time of class 1 when $N = 40$. For LLF, the limiting factor is the loss probability of class 1 when $N \le 12$, the loss probability of class 2 when $12 < N \le 32$ and the mean waiting time of class 1 when $N \ge 34$. In Figure 5, where $2\lambda_1 = \lambda_2$, the limiting factor is the loss probability of class 2 for HOL, the loss probability of class 1 for SLF, the loss probability of class 2 when $N \le 26$ and the mean waiting time of class 1 when $N \ge 28$ for LLF.

The second set of examples (Figures 6 to 11), shows how loss probabilities and mean waiting times of two classes vary with the total load. Again, three service disciplines and three load mixes are used. In all these examples, $N = N_1 = 40$ is assumed and the total load changes from 0.05 to 0.95. Figures 6 to 8 are curves of loss probabilities versus total load with $\lambda_1 = \lambda_2$ in Figure 6, $\lambda_1 = 2\lambda_2$ in Figure 7 and $2\lambda_1 = \lambda_2$ in Figure 8. Figures 9 to 11 are curves of mean waiting times versus total load with $\lambda_1 = \lambda_2$ in Figure 9, $\lambda_1 = 2\lambda_2$ in Figure 10 and $2\lambda_1 = \lambda_2$ in Figure 11.

The loss probabilities of LLF surprisingly resemble the loss probabilities of HOL in all three figures. On the other hand, the mean waiting times of HOL and LLF are in opposite directions. HOL tends to minimize the mean waiting time of class 1 and maximize the mean waiting time of class 2 while LLF tends to equalize the two. Indeed, this resembling and contrast character between HOL and LLF holds also in the next set of numerical examples when $N \approx N_1$ and can be explained intuitively. It seems that loss probabilities and mean waiting times of HOL are least sensitive to the change of the ratio of λ_1 and λ_2 for a given total load, while LLF and SLF are more and most sensitive.

The last set of examples (Figures 12 to 17), shows changes of loss probabilities and mean waiting times of two classes with the increase of N_1. In all these examples, $N = 40$ and $\rho = 0.9$ are assumed. Figures 12 to 14 are curves of loss probabilities versus N_1 with $\lambda_1 = \lambda_2$ in Figure 12, $\lambda_1 = 2\lambda_2$ in Figure 13 and $2\lambda_1 = \lambda_2$ in Figure 14. Figures 15 to 17 are curves of mean waiting times versus N_1 with $\lambda_1 = \lambda_2$ in Figure 15, $\lambda_1 = 2\lambda_2$ in Figure 16 and $2\lambda_1 = \lambda_2$ in Figure 17.

These examples show that once N_1 surpasses certain value, it no longer significantly affects the loss probabilities and mean waiting times.

9 Conclusions

In this work, we analyzed a queueing model $M_1, M_2/G_1, G_2/N$ with different scheduling and push-out schemes. Our work can be used to evaluate the performance of an output link of ATM switches with two-class priority traffics and may also have other applications in computer and communications systems.

By introducing the function α, we were able to consider various scheduling disciplines such as HOL, SLF, LLF and Random Scheduling. By dividing the total buffer spaces into two parts, we created a push-out scheme that permits a controlled share of the buffer spaces between two classes. We gave an exact solution for loss probabilities of both classes, the queue length distribution and mean waiting time for class 1. An approximate solution for the queue length distribution and mean waiting time for class 2 was also obtained. We gave a set of numerical examples which consider the loss probabilities and mean waiting time simultaneously. It remains to extend these results to cases of bursty arrivals.

REFERENCES

[1] CCITT Recommendation I.121: On the Broadband Aspects of ISDN. *CCITT Blue Book*, Geneva, 1989.

[2] CCITT Draft Recommendation I.361: ATM Layer Specification for B-ISDN. *Study Group XVIII*, Geneva, January, 1990.

[3] B. T. Doshi and H. Heffes: Overload Performance of Several Processor Queueing Disciplines for the M/M/1 Queue. *IEEE Transactions on Communications*, Vol. COM-34, No. 6, June 1986, pp. 538-546.

[4] S. Sumita and T. Ozawa: Achievability of Performance Objectives in ATM Switching Nodes. *Proceedings of the International Seminar on Performance of Distributed and Parallel Systems*, Kyoto, December 7-9, 1988, p. 45-56.

[5] S. Sumita: Synthesis of an Output Buffer Management Schemes in a Switching System for Multimedia Communications. *IEEE INFOCOM'90*, San Francisco, June 1990, pp. 1226-1233.

[6] G. Hebuterne and A. Gravey: A Space Priority Queueing Mechanism for Multiplexing ATM Channels. *Computer Network and ISDN Systems*, Vol. 20, No. 1-5, December 1990, pp 37-43.

[7] H. Kroner: Comparative Performance Study of Space Priority Mechanisms for ATM Networks. *IEEE INFOCOM'90*, San Francisco, June 1990, pp. 1136-1143.

[8] T. Takenaka: Buffer Management Schemes for a Heterogeneous Packet Switching System. *Trans. IECE Japan*, J67-B, 6, pp. 505-512, 1984. (in Japanese)

[9] R. W. Wolff: Poisson Arrivals See Time Averages. *Operations Research*, Vol. 30, No. 2, April 1982, pp 223-231.

[10] Hiroshi Saito: Queueing Analysis of a Cell Loss Probability Control in ATM Networks. *Proceedings of the International Teletraffic Congress*, Copenhagen, 1991.

[11] D. P. Heyman: The Push-out Priority Queue Discipline. *Operations Research*, Vol. 33, 2, p. 397-403.

APPENDIX

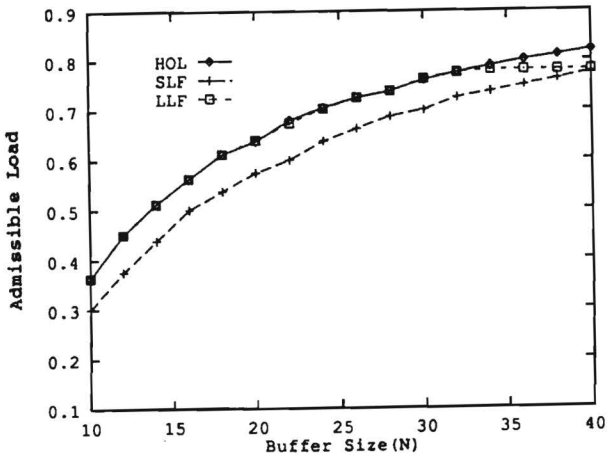


Figure 3: Admissible Load versus Buffer Size ($\lambda_1 = \lambda_2$)

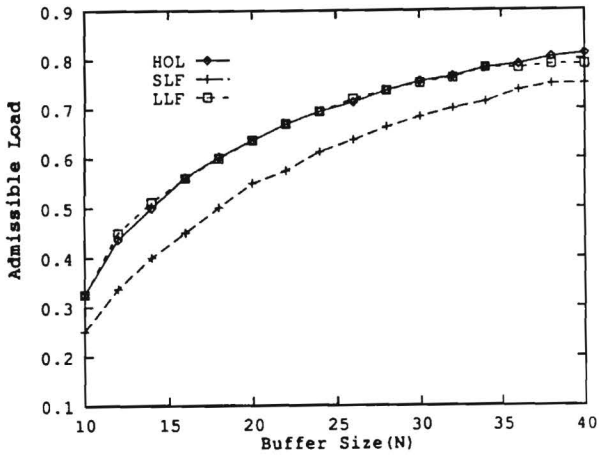


Figure 4: Admissible Load versus Buffer Size ($\lambda_1 = 2\lambda_2$)

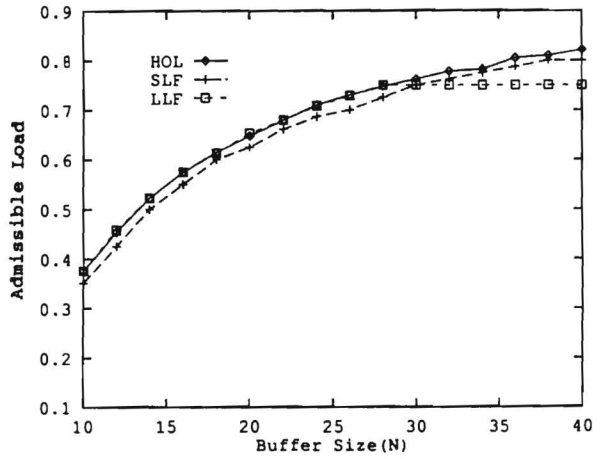


Figure 5: Admissible Load versus Buffer Size ($2\lambda_1 = \lambda_2$)

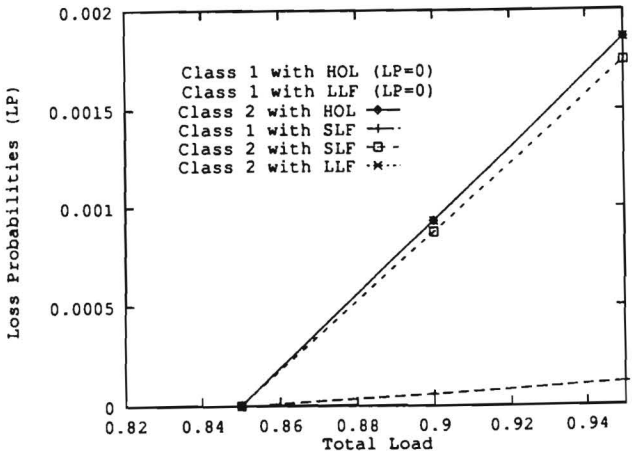


Figure 6: Load versus Loss Probabilities ($N = N_1 = 40, \lambda_1 = \lambda_2$)

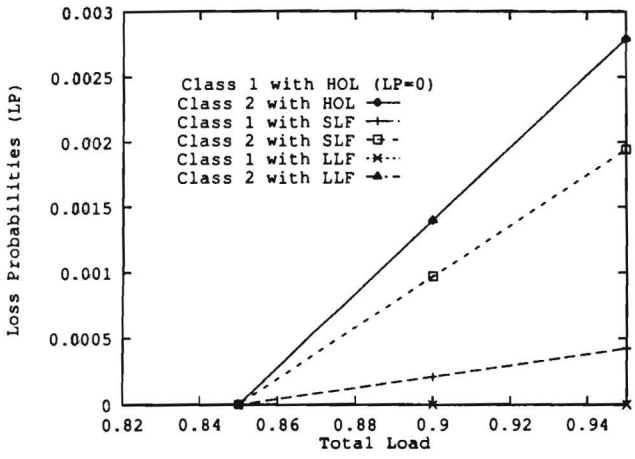


Figure 7: Load versus Loss Probabilities ($N = N_1 = 40, \lambda_1 = 2\lambda_2$)

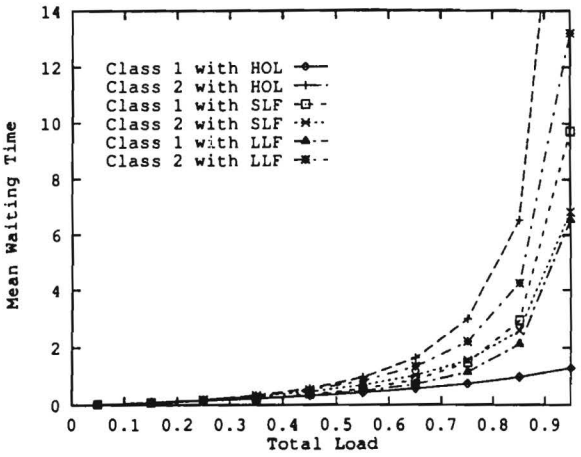


Figure 10: Load versus Mean Waiting Time ($N = N_1 = 40, \lambda_1 = 2\lambda_2$)

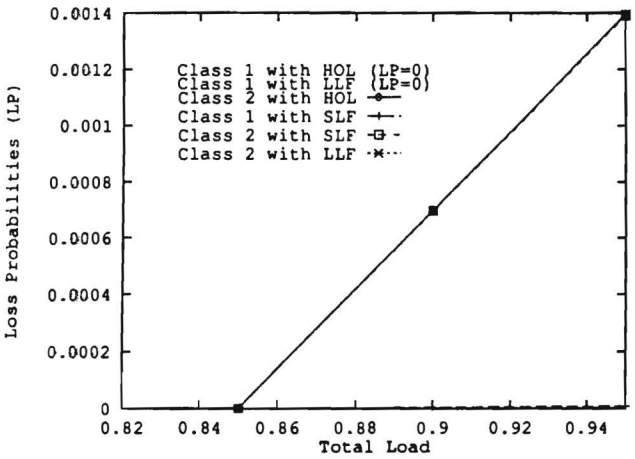


Figure 8: Load versus Loss Probabilities ($N = N_1 = 40, 2\lambda_1 = \lambda_2$)

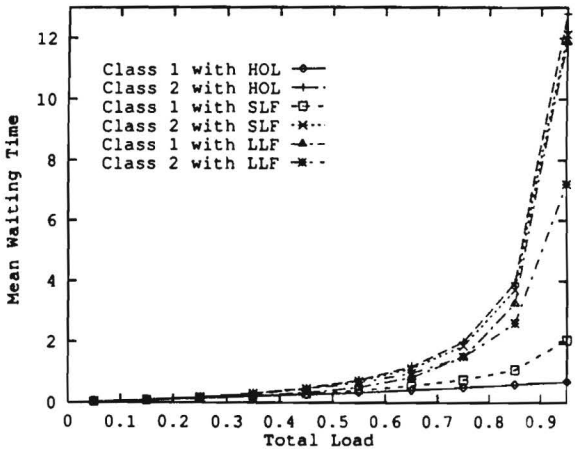


Figure 11: Load versus Mean Waiting Time ($N = N_1 = 40, 2\lambda_1 = \lambda_2$)

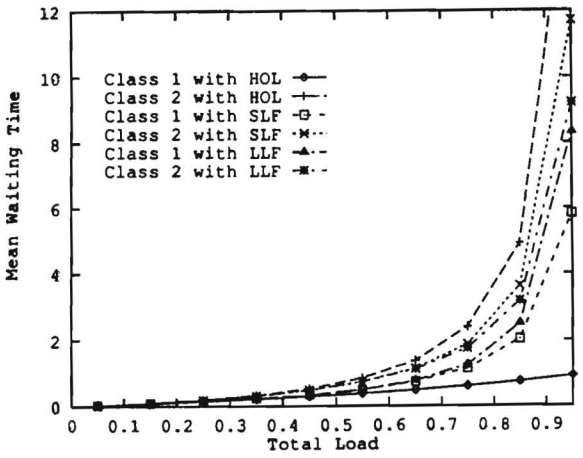


Figure 9: Load versus Mean Waiting Time ($N = N_1 = 40, \lambda_1 = \lambda_2$)

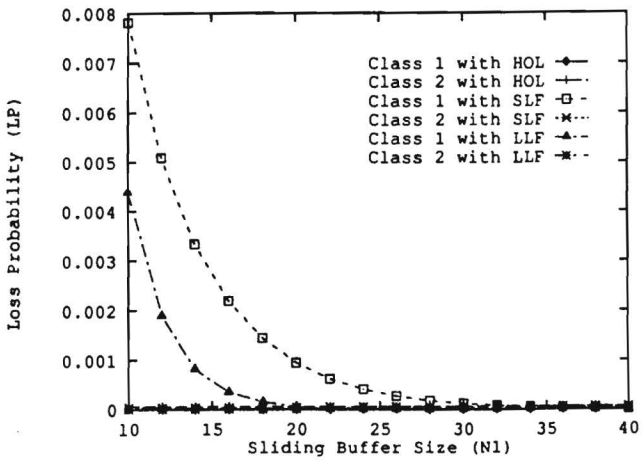


Figure 12: Sliding Buffer Size (N_1) versus Loss Probabilities ($N = 40, \lambda_1 = \lambda_2$, Total load = 0.9)

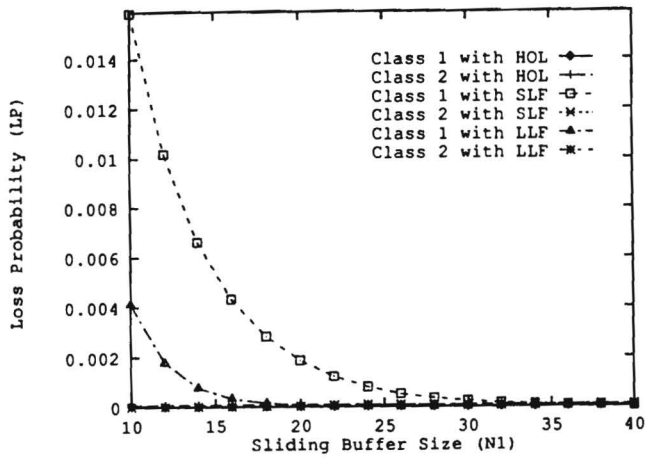


Figure 13: Sliding Buffer Size (N_1) versus Loss Probabilities
($N = 40, \lambda_1 = 2\lambda_2$, Total load = 0.9)

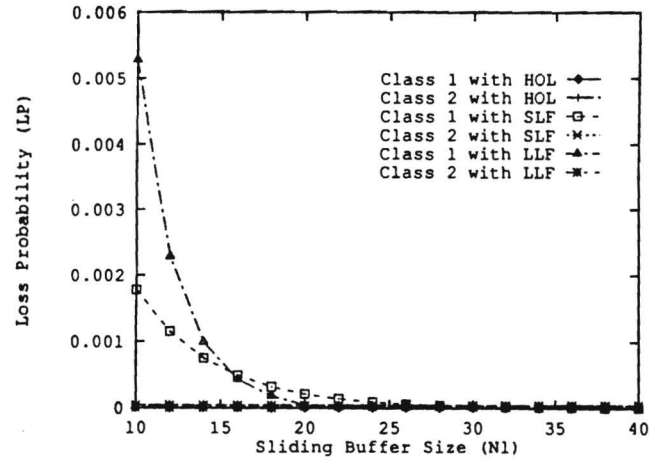


Figure 14: Sliding Buffer Size (N_1) versus Loss Probabilities
($N = 40, 2\lambda_1 = \lambda_2$, Total load = 0.9)

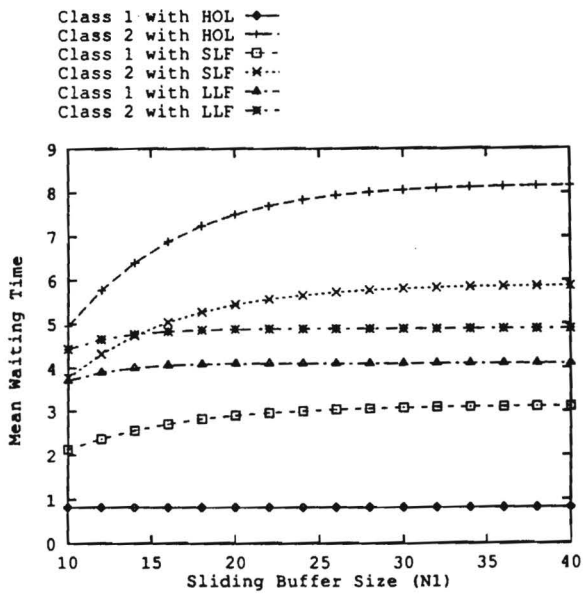


Figure 15: Sliding Buffer Size (N_1) versus Mean Waiting Time
($N = 40, \lambda_1 = \lambda_2$, Total load = 0.9)

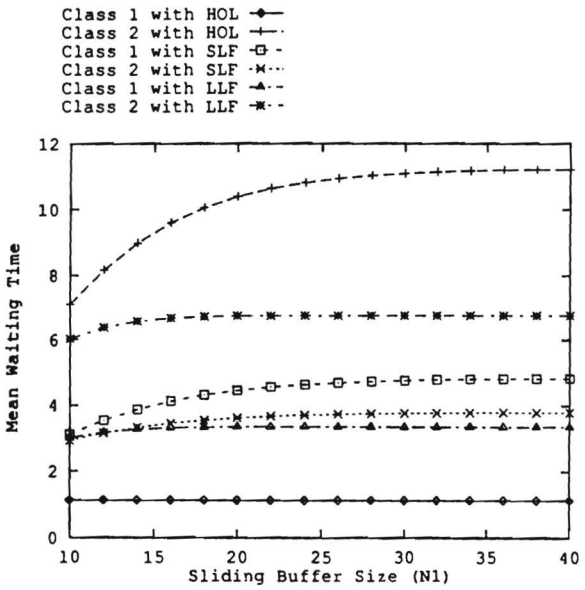


Figure 16: Sliding Buffer Size (N_1) versus Mean Waiting Time
($N = 40, \lambda_1 = 2\lambda_2$, Total load = 0.9)

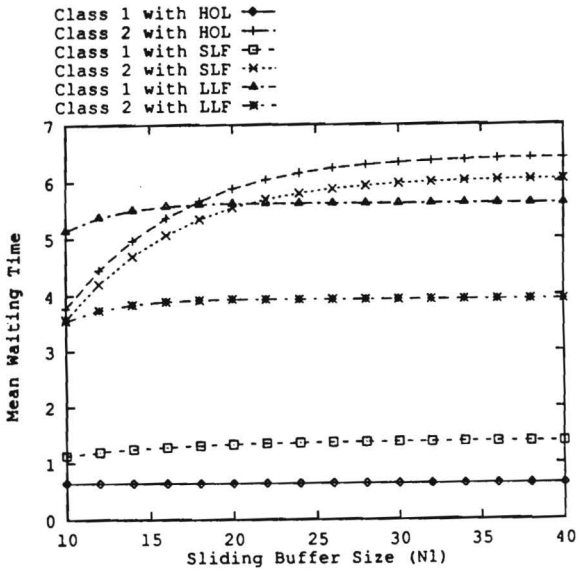


Figure 17: Sliding Buffer Size (N_1) versus Mean Waiting Time
($N = 40, 2\lambda_1 = \lambda_2$, Total load = 0.9)

Queueing Networks with Finite Capacity
R.O. Onvural and I.F. Akyildiz (Editors)

Exact Analysis of Multiple Job Classes and Different Types of Blocking

T. Choukri

College of Computing, Georgia Institute of Technology, Atlanta, GA 30332, USA

Abstract

Queueing networks with finite buffers can be used to model communication, computer and manufacturing systems. We prove that a multi-job class open network with finite capacity queues is *pathwise-equivalent* to a closed queueing network with finite buffers. This implies particularly that the two networks have the same queue length distribution. Furthermore, the service and interarrival time distributions are assumed to be general. We prove that such networks are irreducible if the underlying networks with infinite capacities are irreducible. Finally, the steady-state distribution of a multi-job class queueing network with finite buffers is computed using the steady-state distribution of a single job class queueing network with finite buffers. Moreover, we derive the performance measures for the individual classes using only the global performance measures of the constructed queueing network. [1]

1 Introduction

Queueing networks with finite capacity queues are useful to model communication, computer and manufacturing systems [7, 8, 13]. Different types of blocking have been studied in the literature: *blocking-after-service, blocking-before-service, and repetitive-service with random or fixed destination* [2, 7, 9, 13]. Some equivalencies between queueing networks with different blocking mechanisms have been established in terms of steady-state queue length distribution [8, 9, 14]. Also some algorithms have been developed to compute performance measures in these networks [1, 2, 3, 5, 10, 11, 12]. However, the majority of the results is applicable only to single job class queueing networks with blocking. The major goal of this paper is to develop a tool to extend the known results for single job class queueing networks to multi-job class queueing networks. The steady-state distribution of a multi-job class queueing network with finite buffers is computed using the steady-state distribution of a single job class queueing network with finite buffers. Moreover, we derive the performance measures for the individual classes using only the global performance measures of the constructed single job class queueing network with finite buffers. This is an extension of the result given in [4] for BCMP networks. However the proof in [4] is based explicitly on the product-form solution for multi-job class and single job class networks. However, such proof cannot be extended to finite buffer queueing networks since a general solution does not exist currently. The constructed queueing network can easily be analyzed and performance measures for each station can be obtained by applying any single job class computational algorithm [1, 2, 3, 5, 10, 11, 12]. We prove also that a multi-job class open network with finite capacity queues is *pathwise-*

[1]This work was supported by National Science Foundation (NSF) under Grant No. CCR-90-11981.

equivalent to a closed queueing network with finite buffers. This implies particularly that the two networks have the same queue length distribution. Furthermore, the service and interarrival time distributions are assumed to be general. This extends the results given in [14] for an exponential single job class queueing network with finite capacities. This extension is based on the exploitation of the notion of residual service time distribution in spite of memoryless of exponential distributions. We prove also that such networks are irreducible if the underlying networks with infinite capacities are irreducible.

The paper is organized as follows. In section 2 we prove the equivalencies between open and closed multi-job class queueing networks with finite buffers and general service time distributions. In section 3 we extend the result about the irreducibility property in queueing networks with infinite buffers to queueing networks with finite buffers. In section 4 we construct the single job class queueing network. We show how the routing probability matrix can be determined. We state a theorem showing that both queueing networks with single job class and multiple job classes have the same queue length steady-state distribution. We show also that the steady state distribution of the multi-job class queueing network is equivalent to the queue length steady-state distribution of the single job class queueing network up to a multiplicative terms depending only on the routing probability matrix. We introduce new formulae for the computation of performance measures for individual classes in section 5. Section 6 concludes the paper.

2 Equivalencies between Open and Closed Networks with Finite Buffers

In [14], Onvural and Perros show that a single job class open queueing network with finite capacity queues is equivalent to a closed queueing network with blocking. They assume that the interarrival and service time distributions are exponential. The proof is essentially based on the memoryless property of the exponential distribution. In the following we extend this result to multi-job class queueing networks with blocking and to general interarrival and service time distributions using the notion of residual service time distribution.

We consider a multiple job class queueing network with N stations and R classes. Jobs may change class membership according to the routing probabilities $[P] = (p_{ir,js})$ where $p_{ir,js}$, for $1 \leq i,j \leq N$; $1 \leq r,s \leq R$, is the probability that a job of class r completes service at station i proceeds to station j and joins class s. The probability that an external arrival first enters station j in class s is denoted by $p_{0,js}$ and the probability that a job of class r leaves the network after receiving the service at station i is denoted by $p_{ir,0}$. The transition matrix $[P]$ defines a Markovian process with states identified by (i,r). The Markov process is assumed to be decomposable into C ergodic chains, denoted by $C_1,\ldots,C_C$. We denote C_{ic} as the set of classes belonging to chain c, $(1 \leq c \leq C)$ at node i, $(1 \leq i \leq N)$. We denote also C_{open} as the set of open chains. The number of open chains will be denoted by CO. The terms $p_{0,js}$ and $p_{ir,0}$ for closed chains will be equal to zero.

We assume that for each open chain c, the jobs arrive according to a general distribution G_0^c. The service time distribution is assumed to be general and is denoted by G_i^r for class r in station i. For the buffer capacities, we consider two types of stations:

- A station i is of type CB (common buffer) if there is a common buffer with capacity B_i shared by all job classes in this station.
- A station i is of type SB if there is a specific buffer for each job class with capacities $(B_i^r)_{r \in C_i}$.

The total capacity of a station i is equal to the buffer capacities plus the number of servers denoted by m_i. The blocking mechanism for each station may be one of the following types:

- Repetitive-service-blocking with fixed destination
- Repetitive-service-blocking with random destination
- Blocking-after-service
- Blocking-before-service

In the following this network will be referred to as Γ. We assume that Γ is deadlock free [6] and all the buffer capacities $(B_i)_{i\in CB}$, $(B_i^r)_{i\in SB}$ are finite. Hereafter, we construct a closed queueing network referred to as Θ *pathwise-equivalent* to Γ: we say that two networks are *pathwise-equivalent* if there is one-to-one correspondence between their trajectories.

Theorem 2.1 *The queueing network Γ is pathwise-equivalent to a closed queueing network Θ composed of $N+CO$ stations. The N^{th} first stations in Θ have the same characteristics as the N^{th} first stations in Γ. The stations enumerated $N+1,\ldots,N+CO$ in Θ are of type CB and contain a single server each. The blocking mechanism for these stations is repetitive-service-with-random-destination, the service discipline is FCFS, and the service time distributions are $G^*_{N+c} = G_0^c$ $(1 \le c \le CO)$. The routing probability from a station $N+c$, $(1 \le c \le CO)$ is*

$$p^*_{N+c,ir} = p_{0,ir}, \quad r \in C_{ic}$$

The routing probability to a station $N+c$, $(1 \le c \le CO)$ is

$$p^*_{ir,N+c} = p_{ir,0}, \quad r \in C_{ic}$$

The service time distributions, the blocking mechanism and the routing probabilities for the other stations are identical in both networks. The number of jobs in Θ corresponding to an open chain c in Γ is

$$K_c = \sum_{i\in SB}\sum_{r\in C_{ic}} B_i^r + \sum_{i\in CB \& c\in C_i} B_i + \sum_{c\in C_i} m_i + 1, \quad c \in C_{open}. \qquad (1)$$

Proof.
In network Γ a source for an open chain c behaves as a server with a queue containing an infinite number of jobs and evolving according to the service time distribution G_0^c. So, the server is always busy. The state of the network Γ can be represented by: for each station $(1 \le i \le N)$ the state representing the jobs in this station and the jobs blocked by this station, and the service residual time distributions of the jobs being served; for each source the residual service time distribution of the job being served. The constructed closed network Θ is described by the state of the stations i $(1 \le i \le N+CO)$. According to the routing probability p^* a station $N+c$ $(1 \le c \le CO)$, contains only jobs belonging to chain c. We identify the stations $1,\ldots,N$ in both networks and the stations $N+1,\ldots,N+CO$ in Θ with the sources in Γ. For an initial and identical

state for the two networks, let us identify each job in Γ with his corresponding job in the network Θ. For the stations 1 to N the evolution of jobs is identical for both networks since the routing probability, the service time distributions, and the blocking mechanisms are the same. There is a particular rule for the station $N+c$ in Θ and the source c in Γ. A job blocked in source c of network Γ is lost. The corresponding job in the station $N+c$ in Θ is also blocked and will receive another service according to the repetitive blocking discipline. We identify again this job with the successor of the lost job in Γ. This means that we identify the job being served in the station $N+c$ in Θ and the job in source c in Γ. This identification can be done because there is always a job in station $N+c$ due to condition (1). Moreover, each station $N+c$ $(1 \leq c \leq CO)$ in Θ can be described only by the residual service time of the job being served (as for the sources in Γ). The number of jobs in these stations can be determined by the total number of jobs in chain c and the number of jobs in chain c in stations $1 \ldots N$.
Q.E.D.

3 Irreducibility

We consider a single job class closed queueing network with N stations, K jobs and exponential service time distributions. The routing probability $P = (p_{ij})_{1 \leq i,j \leq N}$ is assumed to be irreducible. For the infinite buffer capacity case, the vector constituted by the number of jobs in each station is an irreducible Markov chain. This is due to the fact that for any stations i and j, a job can move from station i to station j without any changes in the number of jobs in the other stations. According to the irreducibility of the matrix P, there is at least a sequence $ij_1j_2 \ldots j_kj$ such that $p_{ij_1}p_{j_1j_2} \ldots p_{j_kj} > 0$. Then, a job can move with positive probability from station i to station j following the path $i, j_1, j_2, \ldots, j_k, j$. In the finite capacity network, these transitions may be not possible due to full stations. However, the move from station i to station j may be achieved with respect to the full stations.

Theorem 3.1 *We assume that the network is deadlock free and that the routing probability is irreducible. Then, the Markov process constituted by the number of jobs in each station is irreducible.*

Proof.
The proof is based on three steps. The *first step* is that from any unblocked state U_1 the network can evolve to any other unblocked state U_2. This is due to the fact that a job can be moved from a station to another without any changes in the remaining stations. Consider two stations i and j with at least one job at station i and one free buffer at station j. Since P is irreducible, there exist at least a path $i, j_1, j_2, \ldots, j_k, j$ such that $p_{ij_1}p_{j_1j_2} \ldots p_{j_kj} > 0$. A job from the last station $j^1 (i \leq j^1 \leq j_k)$ with at least one job, before station j can be moved to station j. The rule will be applied recursively to stations $j^1, j^2, \ldots$ until station $i = j^l$, where j^2 $(i \leq j^2 \leq j^1)$ is the last station with at least one job, before station j^1, and so on. At the end of these transitions, there will be one job less in station i, one job more in station j, and the same number of jobs in the intermidiate stations. The *second step* is that there is at least one unblocked state U_3 who may lead to a blocked state B_1. To illustrate this step we consider for instance, a type 1 stations and BAS blocking mechanism queueing network. Let us consider a blocked state $B_1 = (k_1, k_2, \ldots, k_{j_1-1}, (k_{j_1}, f_{j_1}), (k_{j_2}, f_{j_2}), \ldots, (k_{j_L}, f_{j_L}), \ldots, k_N)$ where k_i represents the number of jobs in station i $(1 \leq i \leq N)$, (k_{j_l}, f_{j_l}) represents the number of jobs in station j_l $(1 \leq l \leq L)$ and that this station is blocked by station f_{j_l}. This implies that $p_{j_1f_{j_1}} > 0, p_{j_2f_{j_2}} > 0, \ldots, p_{j_Lf_{j_L}} > 0$. So, the transition from

$U_3 = (k_{1,2}, \ldots, k_{j_1-1}, k_{j_1}, k_{j_2}, \ldots, k_N)$ (unblocked state) to B_1 is possible with positive probability. The *third step* is that any blocked state B_2 can evolve to an unblocked state U_4 due to the deadlock free assumption. Based on these three macro-steps, the network can evolve from any state to any other one.

Q.E.D.

Note that this theorem can be generalized to multiple chains as follows: if the underlying BCMP network contains R Markov chains, then the network with finite buffer capacities and deadlock free condition contains also R Markov chains.

4 Composition of Multi-Job Classes into Single Job Class in Queueing Networks with Finite Buffers

In [4] we prove that the steady-state distribution of a multi-job class BCMP network (single chain) is equivalent to a single job class BCMP network steady-state distribution. Here we extend this result to networks with finite capacity queues.

4.1 Model Description

In this section we consider a multi-job class closed queueing network with N stations and K jobs. The routing matrix, denoted by $P = (p_{ir,js})_{1\leq i,j\leq N,\ 1\leq r,s\leq R}$ is assumed to be irreducible. We assume also that the service does not depend on the job class. The stations have a finite buffer capacities and the blocking mechanism for each station can be either RS-FD, BAS, or BBS. The stations are assumed to be of type CB. We refer to this network as Γ. The class of jobs in station i is described by

$$\bar{K}_i = (k_{i1}, \ldots, k_{ik_i})$$

where k_i is the number of jobs present at station i and $k_{ij} = 1, \ldots, R$ $(1 \leq j \leq k_i)$ is the class index of the j^{th} job in the order of the arrivals. The j^{th} job blocked by station i if any, is described by $n_{ij} = (n^1_{ij}, n^2_{ij})$ where n^1_{ij} is the station index of the j^{th} job blocked by station i and n^2_{ij} is the position of that job in station n^1_{ij}. If station n^1_{ij} is of Type 1, then $n^2_{ij} = 1$ otherwise (i.e. n^1_{ij} is of Type 2, 3, or 4) $1 \leq n^2_{ij} \leq k_{n^1_{ij}}$. If the blocking mechanism of station n^1_{ij} is RS-FD or BBS the destination of the n^2_{ij} job in station n^1_{ij} is station i. We denote by b_i the number of jobs blocked by station i, and by b_{ji} the number of jobs blocked in station j by station i $(b_i = \sum_{j=1}^{N} b_{ij})$. The blocking state of station i is described by the vector

$$\bar{N}_i = (n_{i1}, \ldots, n_{ib_i})$$

The state of station i is defined by $\hat{K}_i = (\bar{K}_i, \bar{N}_i)$, and the state of the entire network is represented by the vector

$$\hat{K} = (\hat{K}_1, \ldots, \hat{K}_N). \tag{2}$$

The description $\hat{K}$ is sufficiently detailed to take into account the class property. The blocked state description $\bar{N}_i$ does not give the class of the blocked jobs by station i to

avoid the redundancy. The class of such job can be determined by the position in its station and the class description $\bar{K}$. As will be shown, the evolution of the number of blocked and unblocked jobs can be determined using only the state-space

$$\hat{k} = (\hat{k}_1, \ldots, \hat{k}_N) \tag{3}$$

where $\hat{k}_i = (k_i, n^1_{i1}, n^1_{i2}, \ldots, n^1_{ib_i})$.
Hereafter, we describe a single job class closed network with the same queue length distribution as Γ. In addition, the steady-state distribution of network Γ is obtained using only the queue length distribution and an argument from combinatorics theory.

4.2 Constructing Single Job Class Queueing Network

An analysis of "to-belong-to-a-class" property is needed in order to study the queue length distribution separately of the classes. For a tagged job α in Γ, we define the Markovian chain $(Y_n, Z_n)_{n\leq 1}$ where Y_n and Z_n are the station and the class indexes of the tagged job at the n^{th} transition. The transition matrix of the Markovian chain is also the routing probability matrix $[P]$. So, the steady-state distribution $(e_{ir})_{1\leq i\leq N, 1\leq r\leq R}$ of $(Y_n, Z_n)_{n\leq 1}$ is the solution of the traffic equation:

$$e_{ir} = \sum_{j=1}^{N}\sum_{s=1}^{R} e_{js}p_{js,ir} \tag{4}$$

$$1 = \sum_{i=1}^{N}\sum_{r=1}^{R} e_{ir}. \tag{5}$$

We denote by e_i and $e_{r/i}$ for the tagged job α, respectively the steady-state distribution to be at station i (just before a transition) and its steady-state distribution to be of class r given that it is at station i, i.e.,

$$e_i = P(Y_n = i) \tag{6}$$

$$e_{r/i} = P(Y_n = i, Z_n = r/Y_n = i). \tag{7}$$

Lemma 4.1 *For a tagged job α, the steady-state distribution e_i to be at station i and its steady-state distribution $e_{r/i}$ to be of class r, given it is at station i are computed from*

$$e_i = \sum_{s=1}^{R} e_{is} \tag{8}$$

$$e_{r/i} = \frac{e_{ir}}{\sum_{s=1}^{R} e_{is}}, \tag{9}$$

where (e_{is}) are computed from equations (4) and (5).

Proof.
This lemma is a direct consequence of the probability measure and the conditional probability definitions as given hereafter.

$$e_i = P(Y_n = i)$$

$$= \sum_{s=1}^{R} P(Y_n = i, Z_n = s)$$

$$= \sum_{s=1}^{R} e_{is}.$$

$$e_{r/i} = P(Y_n = i, Z_n = r / Y_n = i)$$

$$= \frac{P(Y_n = i, Z_n = r)}{P(Y_n = i,)}$$

$$= \frac{e_{ir}}{\sum_{s=1}^{R} e_{is}}.$$

Q.E.D.

Lemma 4.2 *A tagged job is routed from station i to station j, without considering its class with the following probability $[Q] = (q_{ij})_{1 \le i,j \le N}$, where*

$$q_{ij} = \sum_{r=1}^{R} \sum_{s=1}^{R} e_{r/i} p_{ir,js}. \qquad (10)$$

Proof.
By definition, we have

$$q_{ij} = E[Y_{n+1} = j / Y_n = i]$$

where E denotes the expectation. According to the conditional probability definition, we have

$$q_{ij} = \sum_{s=1}^{R} E[Y_{n+1} = j, Z_{n+1} = s / Y_n = i]$$

$$= \sum_{r=1}^{R} \sum_{s=1}^{R} E[Y_{n+1} = j, Z_{n+1} = s / Y_n = i, Z_n = r] \frac{P(Y_n = i, Z_n = r)}{P(Y_n = i)}$$

$$= \sum_{r=1}^{R} \sum_{s=1}^{R} e_{r/i} p_{ir,js}.$$

Q.E.D.

Now we can construct the equivalent single job class queueing network which we denote by Θ. The network Θ contains N stations and K jobs and each station has the same queueing discipline (Type 1,2,3 and 4) and the same blocking mechanism (RS-FD, BBS, and BAS) as in network Γ. The routing probability matrix for network Θ is $[Q] = (q_{ij})$ given by expression (10). The service time distributions and the buffer capacities are identical for both networks.

4.3 Steady-State Distributions

In the following, the performance measures related to the network Γ (respectively network Θ) will be indexed by Γ (respectively Θ). We convene that $Y_i(l)$ is equal to i if the l^{th} job in station i is not blocked and it is equal to the index of the destination station if the job is blocked. $Z_i(l)$ is the class of this job.

Theorem 4.1

[a] The network Γ and the network Θ have the same number of blocked and unblocked jobs steady-state distribution, i.e.

$$P^{\Gamma}(\hat{k}) = P^{\Theta}(\hat{k}). \tag{11}$$

[b] The steady-state distribution of network Γ is given by

$$P^{\Gamma}(\hat{K}) = P^{\Theta}(\hat{k}) \prod_{i=1}^{N} \prod_{l=1}^{k_i} e_{Z_i(l)/Y_i(l)} \tag{12}$$

Proof.
Firstly, the class of a tagged job α in network Γ is governed by the same rules as a single job in the network since the routing probability does not depend on the blocking scheme neither on the number of jobs. For this reason, we excluded RS-RD blocking mechanism from this model. Secondly, the class of each job evolves independently from the others. According to definitions (2) and (3) of the states $\hat{K}$ and $\hat{k}$, we have

$$P^{\Gamma}(\hat{K}) = P^{\Gamma}(\hat{k}_i, Y_i(1), Z_i(1), \ldots, Y_i(k_i), Z_i(k_i), 1 \leq i \leq N)$$

The conditional probability definition gives

$$P^{\Gamma}(\hat{K}) = P^{\Gamma}(\hat{k}) P^{\Gamma}(Y_i(1), Z_i(1), \ldots, Y_i(k_i), Z_i(k_i), 1 \leq i \leq N | \hat{k}, Y_i(1), \ldots, Y_i(k_i), 1 \leq i \leq N)$$

Using the independence property explained in the beginning of the proof, we obtain

$$P^{\Gamma}(\hat{K}) = P^{\Gamma}(\hat{k}) \prod_{i=1}^{N} \prod_{l=1}^{k_i} P^{\Gamma}(Y_i(l), Z_i(l) | Y_i(l))$$

The steady-state distribution of a tagged job to be of certain class given it is in a certain station (see equation (7)) implies:

$$P^{\Gamma}(Y_i(l), Z_i(l) | Y_i(l)) = e_{Z_i(l)/Y_i(l)}$$

Henceforth

$$P^{\Gamma}(\hat{K}) = P^{\Gamma}(\hat{k}) \prod_{i=1}^{N} \prod_{j=1}^{k_i} e_{Z_i(j)/Y_i(j)}$$

All we need now is to prove that $P^{\Gamma}(\hat{k}) = P^{\Theta}(\hat{k})$, which will be done using the global balance equation for both networks. Remark that $(\hat{k})$ is a Markov process in network Θ but not in Γ. For the sake of simplicity, we assume that all stations are of type 1. Three different types of transitions may lead to the state $\hat{K}$ (in network Γ):

[a] Transition from a station to another one: A job moves from station i to station j. We assume in this case that station i is neither blocked nor blocking another station ($b_i = 0$). The set of blocked stations will be denoted by Δ. The initial state has to be $T^{ir,j}(\hat{K})$ where $T^{ir,j}(\hat{K})$ is the same state as $\hat{K}$ but with one more job of class r at the the first position in station i and one less job in station j in the last position. If we denote s the class of the job in the last position in station j for $\hat{K}$ i.e. $s = k_{jk_j}$, This transition occurs with rate $\mu_i(k_i + 1)p_{ir,js}$.

[b] Deblocking transitions: A job of class r in the first postion at station $i_0 = i$ moves in class s to station j. We assume that station i_0 is blocking at least one station. We denote the consecutive blocked stations by $i_1 \ldots, i_l$, i.e. i_1 is the first station blocked by station i_0, i_2 is the first station blocked by station i_1, and so on ($l \geq 1$). The initial state $D^{i_0,i_1\ldots,i_l r,j}(\hat{K})$ that may lead to state $\hat{K}$ is as follows: the state of station j is the same as $\hat{K}_j$ but minus the last job which is in class $s = k_{jk_j}$. The state of station $i_0 = i$ is the same as $\hat{K}_i$ but minus the last job and plus a job in class r in the first position. There is also one more station i_1 blocked by station i in the first position. The state of station $i_g (1 \leq g \leq l-1)$ is the same as $\hat{K}_g$ but minus the last job and plus a job in class $k_{i_g 0} = k_{i_{g-1},k_{i_g}}$ in the first position. For $g \leq l-1$ there is also one more station i_{g+1} blocked by station i_g in the first position. This transition occurs with rate $\mu_i(k_i + 1)p_{ir,js}$.

[c] Blocking transitions: A job of class r in station i would move in class s to a full station j i.e. $k_j = B_j$. So, the job is blocked. In fact, station i is the last blocked station for the state $\hat{K}$ i.e. $i = n^1_{jb_j}$, and s is the class of the first job in station i i.e. $s = k_{i1}$. The initial state has to be $B^{ir,j}(\hat{K})$ where $B^{ir,j}(\hat{K})$ is the same state as $\hat{K}$ but with the first job in station i in class r in spite of class s, and minus the last blocked job in station j. This transition occurs with rate $\mu_i(k_i)p_{ir,js}$.

The global balance equation, according to the transitions described above is given by

$$\begin{aligned} P^{\Gamma}(\hat{K}) \sum_{i \notin \Delta} \mu_i(k_i) \;=\; & \sum_{i \notin \Delta, b_i = 0} \sum_{j s.t. k_j < B_j} \sum_{r=1}^{R} P^{\Gamma}(T^{ir,j}(\hat{K}))\mu_i(k_i+1)p_{ir,js} \\ & + \sum_{i \notin \Delta, b_i > 0} \sum_{j s.t. k_j < B_j} \sum_{r=1}^{R} \sum_{i_1 \ldots i_l} P^{\Gamma}(D^{i_0,i_1\ldots,i_l r,j}(\hat{K}))\mu_i(k_i+1)p_{ir,js} \\ & + \sum_{b_j > 0, i = n^1_{jb_j}} \sum_{r=1}^{R} P^{\Gamma}(B^{ir,j}(\hat{K}))\mu_i(k_i)p_{ir,js} \end{aligned} \tag{13}$$

We derive from the definitions of $\hat{K}$ and $\hat{k}$ and by summing over all the classes that

$$\sum_{classes} P^{\Gamma}(\hat{K}) = P^{\Gamma}(\hat{k}). \tag{14}$$

In a similar way, we obtain by summing over all the classes but class r that

$$\sum_{classes} P^{\Gamma}(T^{ir,j}(\hat{K})) \;=\; P^{\Gamma}(T^{i,j}(\hat{k}))e_{r/i} \tag{15}$$

$$\sum_{classes} P^{\Gamma}(B^{ir,j}(\hat{K})) = P^{\Gamma}(B^{i,j}(\hat{k}))e_{r/i} \quad (16)$$

$$\sum_{classes} P^{\Gamma}(D^{ir,j}(\hat{K})) = P^{\Gamma}(D^{i,j}(\hat{k}))e_{r/i} \quad (17)$$

The notations $T^{i,j}(\hat{k})$, $B^{i,j}(\hat{k}))$ *and* $D^{i,j}(\hat{k}))$ are self-explanatory and define the states that may lead to state $\hat{k}$ in network Θ. We deduce from equations (14)-(17), and from definition (10) of q_{ij} that

$$\begin{aligned} P^{\Gamma}(\hat{k}) \sum_{i \notin \Delta} \mu_i(k_i) &= \sum_{i \notin \Delta, b_i=0} \sum_{j s.t. k_j < B_j} P^{\Gamma}(T^{i,j}(\hat{k}))\mu_i(k_i+1)q_{ij} \\ &+ \sum_{i \notin \Delta, b_i>0} \sum_{j s.t. k_j < B_j} \sum_{i_1 \ldots i_l} P^{\Gamma}(D^{i_0,i_1 \ldots,i_l,j}(\hat{k}))\mu_i(k_i)q_{ij} \\ &++ \sum_{b_j>0, i=n^1_{jb_j}} P^{\Gamma}(B^{i,j}(\hat{k}))\mu_i(k_i)q_{ij} \end{aligned} \quad (18)$$

Equation (18) is in fact the global balance equation of network Θ. Thus by unicity of the solution of such equation, we deduce that

$$P^{\Gamma}(\hat{k}) = P^{\Theta}(\hat{k}).$$

Q.E.D.

In the following we assume that the performance measures such as mean number of unblocked jobs $\bar{h}_i$ and the mean number of jobs $\bar{b}_{ij}$ blocked in station i by station j for network Θ are completely determined. This step may be done using an analytical solution as in [1, 2, 3, 5, 10, 11, 12].

5 Performance Measures for Multi-Job Class Queueing Network

We denote by h_{ir} the number of unblocked jobs of class r in station i and by b_{ijr} the number of jobs of class r blocked in station i by station j. We define also the following rates for the i^{th} station $(1 \leq i \leq N)$

$$\alpha_{irs} = \frac{e_{ir}}{e_{is}} \; for \; 1 \leq r, s \leq R. \quad (19)$$

Theorem 5.1 *The mean number of unblocked jobs $\bar{h}_{ir}$ of class r in station i and the mean number of blocked jobs $\bar{b}_{ijr}$ of class r in station i by station j are computed by*

$$\bar{h}_{i1} = \frac{\bar{h}_i}{\sum_{r=1}^{R} \alpha_{ir1}} \quad (20)$$

$$\bar{h}_{ir} = \alpha_{ir1}\bar{h}_{i1}, \; for \; 1 \leq i \leq N, 1 \leq r \leq R, \quad (21)$$

$$\bar{b}_{ij1} = \frac{\bar{b}_{ij}}{\sum_{r=1}^{R} \alpha_{ir1}} \quad (22)$$

$$\bar{b}_{ijr} = \alpha_{ir1}\bar{b}_{ij1}, \; for \; 1 \leq i \leq N, 1 \leq r \leq R. \quad (23)$$

where $\bar{h}_i$ and $\bar{b}_{ij}$ $(1 \leq i, j \leq N)$ are computed from the network Θ.

Proof.
According to Theorem 4.1 and to combinatorics theory, the steady-state distribution that there are h_{ir} and b_{ijr} jobs regardless of their positions in the queue is a multinomial distribution i.e.,

$$\begin{aligned} P^{\Gamma}(h_{ir}, b_{ijr} \;\; 1 \leq i,j \leq N \;\; 1 \leq r \leq R) \;\; = \;\; & P^{\Theta}(h_i, b_{ij} \;\; 1 \leq i,j \leq N) \\ & \times \prod_{i=1}^{N} \binom{h_{ir} \;\; 1 \leq r \leq R}{h_i} \\ & \times \prod_{i=1}^{N} \binom{b_{ijr} \;\; 1 \leq j \leq N \;\; 1 \leq r \leq R}{b_{ij}} \\ & \times \prod_{i=1}^{N} \prod_{r=1}^{R} (e_{r/i})^{h_{ir}} \prod_{i=1}^{N} \prod_{j=1}^{N} \prod_{r=1}^{R} (e_{r/j})^{b_{jir}} \end{aligned}$$

So the conditional expectation of h_{ir} and b_{ijr} knowing h_l and b_{lm} $1 \leq l,m \leq N$, are obtained by

$$E[h_{ir}/h_l, b_{lm}, 1 \leq l,m \leq N] \;\; = \;\; P^{\Theta}(h_l, b_{lm} \;\; 1 \leq l,m \leq N) h_i e_{r/i} \tag{24}$$

$$E[b_{ijr}/h_l, b_{lm}, 1 \leq l,m \leq N] \;\; = \;\; P^{\Theta}(h_l, b_{lm} \;\; 1 \leq l,m \leq N) b_{ij} e_r/j \tag{25}$$

By computing the expectation over h_i and b_{ij} in equations (24) and (25), we obtain the mean number of blocked and unblocked jobs of class r in station i

$$\begin{aligned} \bar{h}_{ir} \;\; &= \;\; S \bar{h}_i e_{r/i} \\ \bar{b}_{ijr} \;\; &= \;\; T \bar{b}_{ij} e_{r/j} \end{aligned}$$

where S and T are two constants independent of the classes. Thus, we deduce that

$$\frac{\bar{h}_{ir}}{\bar{h}_{i1}} \;\; = \;\; \frac{e_{r/i}}{e_{1/i}} = \alpha_{ir1}, \tag{26}$$

$$\frac{\bar{b}_{ijr}}{\bar{b}_{ij1}} \;\; = \;\; \frac{e_{r/j}}{e_{1/j}} = \alpha_{jr1} \tag{27}$$

These equations combined to the fact that $\bar{h}_i = \sum_{r=1}^{R} \bar{h}_{ir}$ and $\bar{b}_{ij} = \sum_{r=1}^{R} \bar{b}_{ijr}$ imply equations (20)-(23).

Q.E.D.

6 Conclusions

We introduced a new concept which allows to construct a queueing network with finite capacity queues containing a single composite job class. This constructed queueing network model has the same number of blocked and unblocked jobs distribution as the originally given multi-job class queueing network with finite capacity queues. In addition, their steady-state distributions are equal up to a multiplicative term depending only on the routing probability matrix. The performance measures of the multi-job class queueing network with finite buffers are deduced from the performance measures of the single job class queueing networks.

Acknowledgment
I am grateful to Prof. I.F. Akyildiz at Georgia Tech, USA for his suggestions and constructive comments.

References

[1] I. F. Akyildiz. Exact analysis of queueing networks with rejection blocking. *Proc. of the Workshop on Queueing Networks with Blocking, North-Holland*, pages 19–29, 1989.

[2] I. F. Akyildiz. Product form approximations for queueing networks with multiple servers and blocking. *IEEE Transactions on Computers*, 38(1):99–114, Jan. 1989.

[3] I. F. Akyildiz and H. Von Brand. Exact solutions for networks of queues with blocking-after-service. *Technical Report, Georgia Tech, College of Computing*, (GIT-CoC-91-26), April 1991.

[4] I. F. Akyildiz and T. Choukri. Composition of multi-job classes into single job class in multi-chain networks of queues. *Technical Report, Georgia Tech, College of Computing*, Nov. 1991.

[5] I. F. Akyildiz and T. Choukri. Exact product form solution for single class queueing networks with finite capacity and different types of stations. *Technical Report, Georgia Tech, College of Computing*, Nov. 1991.

[6] I. F. Akyildiz and J. Liebeherr. Optimal deadlock free buffer allocation in multiple chain blocking networks of queues. *Proc. of the Int. Conference on the Performance of Distributed Systems and Integrated Communication Networks*, Sep. 10-12 1991.

[7] I. F. Akyildiz and H. G. Perros. Special issue on queueing networks with finite capacity queues: Introduction. *Performance Evaluation*, 10(3), Dec. 1989.

[8] S. Balsamo and G. Iazeolla. Some equivalence properties for queueing networks with and without blocking. *Performance' 83, North-Holland, Amsterdam*, pages 351–360, 1983.

[9] S. Balsamo and V. De Nitto-Parsone. Closed queueing networks with finite capacities: Blocking types, product-form solution and performance indices. *Performance Evaluation*, 12(2):85–102, April 1991.

[10] Y. Dallery and Y. Frein. A decomposition method for the approximate analysis of closed queueing networks with blocking. *Proc. First International Workshop on Queueing with Blocking, North-Holland, Amsterdam*, pages 193–215, 1989.

[11] F.P. Kelly. Reversibility and stochastic networks. *Wiley, Chichester and New York*, 1979.

[12] G. Latouche and M. F. Neuts. Efficient algoritmic solutions to exponential tandem queues with blocking. *SIAM J. ALG. DISC. MATH.*, 1(1):93–106, 1980.

[13] R. O. Onvural. A survey of closed queueing networks with finite buffers. *ACM Computing Surveys*, 22(2):83–121, June 1990.

[14] R. O. Onvural and H. G. Perros. Equivalencies between Closed Queueing Networks with Finite Buffers. *Performance Evaluation*, 9:263–269, 1988/89.

Queueing Networks with Finite Capacity
R.O. Onvural and I.F. Akyildiz (Editors)

Deadlock-Free Buffer Allocations in Multiple Chain Blocking Networks with Tandem Sequences *

Jörg Liebeherr
Computer Science Division,
University of California - Berkeley,
Berkeley, CA 94720, U. S. A.

Abstract

In order to verify that a blocking network with multiple routing chains is deadlock-free, all cycles in the network need to be tested whether they satisfy the deadlock freedom conditions. Networks with so-called tandem sequences are characterized by a high number of cycles. Finding all cycles in a blocking network with tandem sequences is computational impractical if the size of the network is large. It is shown that the effort of finding all cycles can be reduced by substituting each tandem sequence in a blocking network with a single station. If the substitute network is deadlock-free then the original network is deadlock-free. A method is devised that provides a deadlock-free buffer allocation with a minimum total number of buffers for blocking networks with tandem sequences. It is shown that a deadlock-free and minimal buffer allocation for networks consisting of only one tandem sequence can be directly obtained.

1 Introduction

In this work we consider the so-called blocking-after-service (BAS), also referred as type 1 or manufacturing blocking mechanism [1]. A job upon service completion at a station i attempts to enter the destination station j. If station j is full at that moment, the job is forced to wait in station i's server until it can enter destination station j. The server remains blocked for this period of time. It cannot serve other jobs waiting in the queue.

Finite station capacities and blocking can introduce a deadlock situation. In a simple example, a deadlock may occur if a job which has finished its service at station i's server wants to join station j. If station i is full the job is blocked in station i. Another job which has finished service at station j now wants to proceed to station i which is also full. It blocks the jth station. Both jobs are waiting for each other. As a result a deadlock situation arises. There are two possible solutions for the deadlock problem: Either include a strategy to handle deadlocks in the model or simply restrict oneself to cases where deadlock is impossible. We select the second approach and find conditions under which a multiple chain blocking network is deadlock-free. Kundu and Akyildiz [3] showed that queueing networks with a single routing chain are deadlock-free if the number of jobs in the network is less than the capacity of the directed cycle with minimal capacity.

* This work was supported by the National Science Foundation under Grant No. CCR-90-11981

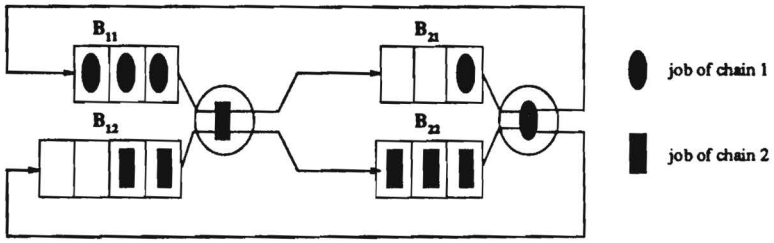


Figure 1: Blocking Network with 2 chains.

However, the conditions for deadlock freedom with multiple routing chains become very complex due to interactions between routing chains.

We demonstrate these interactions in the network model given in Figure 1. There are 2 stations and 2 routing chains in the model. Let B_{ir} denote the buffer of chain r at station i (for $i, r = 1, 2$) and let K_r denote the total number of jobs in chain r ($r = 1, 2$). Let the parameters be given by $K_1 = 5$, $K_2 = 6$, $B_{11} = 3$, $B_{21} = 3$, $B_{12} = 4$ and $B_{22} = 3$. Here a deadlock situation occurs, even though the condition for deadlock freedom given in [3] are satisfied for each routing chain in isolation.

In addition to determining the conditions for deadlock freedom another important issue is the allocation of station capacities in a queueing network such that deadlocks cannot occur. Akyildiz and Liebeherr [2] proved the necessary and sufficient conditions for deadlock freedom in multiple chain queueing networks with blocking. They presented an algorithm that computes a deadlock-free buffer allocation with the least number of total buffers. The algorithm generates the set of so-called buffer cycles. Finally, the deadlock-free allocation with the minimum total number of buffers is obtained by applying linear programming techniques.

However, for certain network topologies it is computationally impractical to compute all buffer cycles. In this study we show for the class of blocking networks with so-called *tandem sequences* that the effort to find a deadlock-free allocation with minimum number of buffers can be reduced.

The remaining sections of this paper are organized as follows. In section 2 we specify the class of queueing networks which is considered in this study. We review the deadlock freedom conditions and the buffer allocation algorithm presented in [2]. In section 3 we are concerned with finding a minimal deadlock-free buffer allocation for blocking networks with tandem sequences. If the entire network constitutes a single tandem sequence we prove that a minimal deadlock-free buffer allocation can be directly obtained. We show that for blocking networks with tandem sequences a minimal deadlock-free buffer allocation can be obtained from a substitute network with a simpler structure. In section 4, we present an example that applies our results. We summarize our results in section 5.

2 Deadlock Freedom in Multiple Chain Blocking Networks

2.1 Model Description

We consider a closed queueing network Γ with the following properties:

a) The system consists of N stations and R disjoint routing chains. Stations and routing chains are referred to by their indexes. Let $\mathcal{N}$ and $\mathcal{R}$ denote the set of stations and routing chains, respectively. Let $\mathcal{R}_i \subseteq \mathcal{R}$ denote the set of routing chains visiting station i. Let $\mathcal{N}_r \subseteq \mathcal{N}$ denote the set of stations visited by routing chain r.

b) Each station i has a single server. The service time distribution and the scheduling discipline of a station is arbitrary, but non-preemptive.

c) Each station keeps separate buffers for jobs from different routing chains. B_{ir} denotes the buffer at station i (excluding the server) for jobs from routing chain r ($i \in \mathcal{N}$, $r \in \mathcal{R}_i$). Each buffer may accommodate only a finite number of jobs. Let Φ be an assignment of capacities to the buffers of Γ, i.e.,

$$\Phi : \{B_{ir} | i \in \mathcal{N},\ r \in \mathcal{R}_i\} \rightarrow \{0, 1, 2, \ldots\ldots\} \tag{1}$$

Note that a buffer can have infinite capacity ($\Phi(B_{ir}) = \infty$) or no capacity at all ($\Phi(B_{ir}) = 0$).

d) The number of jobs in the system is fixed at

$$\underline{K} = (K_1, K_2, \ldots, K_R) \tag{2}$$

where

$$K_r = \sum_{i \in \mathcal{N}_r} k_{ir}, \qquad \text{for } r = 1, \ldots, R \tag{3}$$

represents the total number of jobs in routing chain r, and k_{ir} is the number of jobs of chain r in the i-th station (in buffer and server). The state of the network is represented by a vector $\underline{k} = (\underline{k}_1, \underline{k}_2, \ldots, \underline{k}_N)$ where $\underline{k}_i$ is a vector $(k_{ir_1}, k_{ir_2}, \ldots, k_{ir_{|\mathcal{R}_i|}})$ with $r_u \in \mathcal{R}_i$ ($u = 1, 2, \ldots, |\mathcal{R}_i|$). The total capacity of station i is computed by $\sum_{r \in \mathcal{R}_i} \Phi(B_{ir}) + 1$.

e) A job of routing chain r which has received service in station i proceeds to the station j with probability $p_{ij,r}$ (for $i, j \in \mathcal{N}$, $r \in \mathcal{R}_i$ and $r \in \mathcal{R}_j$). The number of jobs in the buffer of station j for jobs of chain r cannot exceed its capacity $\Phi(B_{jr})$. If B_{jr} is *saturated*, i.e., $k_{jr} = \Phi(B_{jr})$, the job of chain r is blocked at buffer B_{jr} and has to remain in the server of station i until a place in B_{jr} becomes available. This is the BAS (blocking-after-service) blocking mechanism.

2.2 Deadlock Freedom Conditions

The following definitions are needed to state the conditions for deadlock freedom of a network Γ.

Definition 1 *A buffer cycle (of length $t-1$ $(1 \leq t-1 \leq |\mathcal{N}|)$) is a sequence of buffers $C = (B_{\sigma_1\rho_1}, B_{\sigma_2\rho_1}, \ldots, B_{\sigma_{t-1}\rho_{t-1}}, B_{\sigma_t\rho_t})$ such that*

$$\begin{aligned} &\sigma_u \in \mathcal{N} && (1 \leq u \leq t) \\ &\sigma_t = \sigma_1 && \\ &\rho_u \in R_{\sigma_u},\ \ \rho_{u+1} \in R_{\sigma_u},\ \ p_{\sigma_u\sigma_{u+1},\rho_{u+1}} > 0 && (1 \leq u < t). \end{aligned} \tag{4}$$

Let $\mathcal{C}$ denote the set of all buffer cycles in Γ.

Definition 2 *The set of stations with buffers from routing chain r in buffer cycle C is defined by*

$$S_r^{(C)} = \{i \mid B_{ir} \in C, i \in \mathcal{N}, r \in \mathcal{R}_i\} \tag{5}$$

The necessary and sufficient deadlock freedom conditions for multiple chain blocking networks are given by the following theorem [2].

Theorem 1 (DLF condition) *A multiple chain queueing network Γ is* deadlock-free *(DLF) if and only if for all $C \in \mathcal{C}$ with $C = (B_{\sigma_1\rho_1}, B_{\sigma_2\rho_1}, \ldots, B_{\sigma_{t-1}\rho_{t-1}}, B_{\sigma_t\rho_t})$ there exists a routing chain $r \in \mathcal{R}$ such that:*

$$\sum_{1<u\leq t \wedge \sigma_u \in S_r^{(C)}} \Phi(B_{\sigma_u r}) + |S_r^{(C)}| > K_r \tag{6}$$

In other words, each buffer cycle must contain at least one routing chain which cannot both saturate the buffers of its chain in the cycle and have blocked jobs waiting at the saturated buffers. If the DLF condition (equation (6)) is satisfied for a buffer cycle C they are clearly satisfied for all cycles which include the buffers of cycle C as a subset. Therefore, it is not required to test the DLF condition for all buffer cycles to guarantee deadlock freedom of the network. This is formalized in the following Lemma [2].

Lemma 1 *Given a multiple chain queueing network Γ and the set of all buffer cycles $\mathcal{C}$ of Γ. For any $C', C'' \in \mathcal{C}$ let $C' \subseteq C''$, if the set of buffers in C' is a subset of the set of buffers in C''. Let $\tilde{\mathcal{C}} \subseteq \mathcal{C}$ be such that*

$$(\forall C', C'' \in \tilde{\mathcal{C}})(C' \not\subseteq C''). \tag{7}$$

Then, Γ is deadlock-free if the DLF condition (equation (6)) is satisfied for all $C \in \tilde{\mathcal{C}}$.

2.3 Deadlock-Free Buffer Allocation Algorithm

Once the conditions for a deadlock-free network are known, the problem of finding a deadlock-free allocation with the least number of buffers can be investigated. We define *optimal buffer allocations* to be deadlock-free buffer allocations with the least total number of buffers.

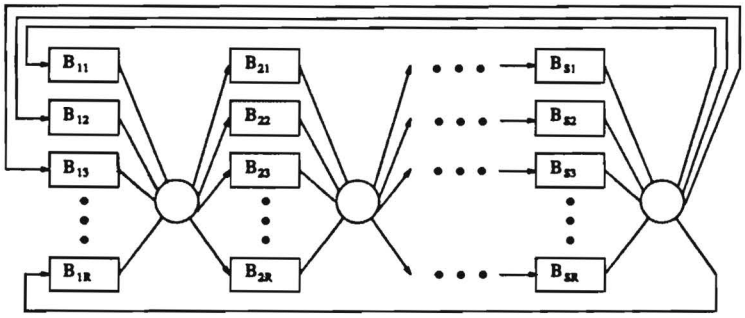


Figure 2: Tandem Network.

Definition 3 *A buffer allocation Φ^* for a queueing network Γ is optimal if:*

(i) Φ^* satisfies the DLF condition (equation (6)) for all $C \in \mathcal{C}$,

(ii) Φ^* is minimal, i.e., no deadlock-free allocation Φ for Γ allocates less capacities to the buffers of Γ than Φ^*:

$$\sum_{i \in \mathcal{N}} \sum_{r \in \mathcal{R}_i} \Phi^*(B_{ir}) \leq \sum_{i \in \mathcal{N}} \sum_{r \in \mathcal{R}_i} \Phi(B_{ir}) \tag{8}$$

In [2], an algorithm was presented that gives an optimal buffer allocation for a blocking network Γ with K_r jobs in each chain r. The computational effort to obtain an optimal buffer allocation is dependent on the number of buffer cycles in the network. In the following section we show that the optimization procedure can be made computationally efficient even if the number of cycles in the network grows large.

3 Special Network Topologies

Even though the optimal buffer allocation algorithm is applicable to all network topologies, the computational complexity of the algorithm increases with the size of $\tilde{\mathcal{C}}$, the set of cycles where no cycle is a subset of another. For some networks, the size of $\tilde{\mathcal{C}}$ can make the optimization algorithm impractical. As a worst case consider the network in Figure 2. Here, $\tilde{\mathcal{C}}$ is identical with the set of all buffer cycles in the network. Note that in the network in Figure 2 the set $\tilde{\mathcal{C}}$ contains R^S cycles. In the following we show that obtaining an optimal buffer allocation can be simplified, if we take advantage of structural properties of the network. Here we consider two types of networks, so-called *tandem networks* which have a topology as shown in Figure 2, and networks with so-called *tandem sequences*, i.e., networks that contain tandem networks as subnetworks.

Definition 4 *Given a multiple chain queueing network* Γ.

1. *A* tandem sequence *is a sequence of stations* $(\kappa_1, \kappa_2, \ldots, \kappa_{s-1}, \kappa_s)$ *such that*

$$\begin{array}{ll} \mathcal{R}_{\kappa_v} = \mathcal{R}_{\kappa_w} & (1 \le v, w \le s) \\ (\forall r \in \mathcal{R}_{\kappa_v})(p_{\kappa_v \kappa_{v+1}, r} = 1) & (1 \le v < s) \\ (\forall r \in \mathcal{R}_{\kappa_v})(p_{i \kappa_{v+1}, r} = 0) & (1 \le v < s, i \ne \kappa_v). \end{array} \tag{9}$$

2. *A* tandem network Γ *with* S *stations is a tandem sequence* $(\kappa_1, \kappa_2, \ldots, \kappa_S, \kappa_{S+1})$ *where* $\kappa_1 = \kappa_{S+1}$.

Next we show that finding an optimal buffer allocation can be simplified if the network is a tandem network or contains tandem sequences. In the next subsection we prove that optimal buffer allocations for tandem networks with blocking can be obtained without running any optimization procedure. Then we prove for blocking networks with tandem sequences that the efforts to find an optimal buffer allocation can be greatly reduced.

3.1 Optimal Buffer Allocation for Closed Tandem Networks

In the next Lemma we show that an optimal buffer allocation for a tandem network assigns nonzero buffer spaces to the buffers of only one station.

Lemma 2 *For a tandem network* Γ *with* S *stations and* R *routing chains, the following buffer allocations* $\hat{\Phi}_i$ $(i \in \mathcal{N})$ *are optimal:*

$$\hat{\Phi}_i(B_{jr}) = \begin{cases} K_r & \text{if } j = i \text{ and } \sum_{\rho \in \mathcal{R} \setminus \{r\}} K_\rho \ge S \text{ and } \sum_{r=1}^{R} K_r \ge S \\ K_r - S + \sum_{\rho \in \mathcal{R} \setminus \{r\}} K_\rho & \text{if } j = i \text{ and } \sum_{\rho \in \mathcal{R} \setminus \{r\}} K_\rho < S \text{ and } \sum_{r=1}^{R} K_r \ge S \\ 0 & \text{if } j = i \text{ and } \sum_{r=1}^{R} K_r < S \\ 0 & \text{if } j \ne i \\ & \quad \text{for } j \in \mathcal{N} \text{ and } r \in \mathcal{R}. \end{cases} \tag{10}$$

Proof:

(1) $\hat{\Phi}_i$ *is a deadlock-free allocation*

Let us first consider the case: $\sum_{r=1}^{R} K_r < S$. In this case, no state exists where all stations of Γ are occupied. Hence, for each cycle C there exists a chain $r' \in \mathcal{R}$ with:

$$|S_{r'}^{(C)}| > K_{r'}, \tag{11}$$

which satisfies equation (6). For the other cases, consider $i = \kappa$ $(1 \le \kappa \le S)$. In a tandem network, each buffer cycle C has to contain a buffer from station κ, say $B_{\kappa\tau}$ $(\tau \in \mathcal{R})$. Since:

$$\begin{array}{lll} \hat{\Phi}_\kappa(B_{\kappa\tau}) = K_\tau & \wedge \ |S_\tau^{(C)}| > 0, & \text{if } \sum_{\rho \in \mathcal{R} \setminus \{\tau\}} K_\rho \ge S, \\ \hat{\Phi}_\kappa(B_{\kappa\tau}) = K_\tau - \sum_{\rho \in \mathcal{R} \setminus \{\tau\}} K_\rho & \wedge \ |S_\tau^{(C)}| > S - \sum_{\rho \in \mathcal{R} \setminus \{\tau\}} K_\rho, & \text{if } \sum_{\rho \in \mathcal{R} \setminus \{\tau\}} K_\rho < S. \end{array} \tag{12}$$

We obtain for either case:

$$\hat{\Phi}_\kappa(B_{\kappa\tau}) + |S_\tau^{(C)}| > K_\tau. \tag{13}$$

This satisfies the DLF condition for chain τ in Cycle C.

(2) $\hat{\Phi}_i$ is minimal
Since buffer allocation $\hat{\Phi}_i$ is a deadlock-free allocation we can use it as an upper bound for any minimal allocation. To prove the minimality of allocations $\hat{\Phi}_i$ for chain r ($r \in \mathcal{R}$) we will show that any other allocation which allocates the same total capacities to the buffers as $\hat{\Phi}_i$ will contain a deadlock. We have to distinguish four cases:

(a) $\sum_{r=1}^{R} K_r < S$
In this case, no capacity is allocated to any buffer, i.e., $\hat{\Phi}_i$ is minimal.

(b) $K_r \geq S$ and $\sum_{\rho \in \mathcal{R} \setminus \{r\}} K_\rho \geq S$
Assume there exists an allocation Φ^* different from any $\hat{\Phi}_i$ with $\sum_{i \in \mathcal{N}_r} \Phi^*(B_{ir}) = K_r$. Without loss of generality we may assume that $R = 2$ with two chains r and r'. Let m ($m < S$) be such that m buffers in chain r $(B_{\kappa_1 r}, B_{\kappa_2 r}, \ldots, B_{\kappa_m r})$ satisfy:

$$\sum_{s=1}^{m} \Phi^*(B_{\kappa_s r}) + m > K_r. \tag{14}$$

Since $K_r > S$ we have:

$$\sum_{s=m+1}^{S} \Phi^*(B_{\kappa_s r}) + (S - m) < K_r, \tag{15}$$

and $\sum_{s=1}^{S} \Phi^*(B_{\kappa_s r}) \leq K_r$ implies:

$$\sum_{s=m+1}^{S} \Phi^*(B_{\kappa_s r}) < m. \tag{16}$$

Because of (15) chain r' must satisfy:

$$\sum_{s=1}^{m} \Phi^*(B_{\kappa_s r'}) + m > K_{r'}. \tag{17}$$

This allows to perform an induction over m.

Induction start: $m = 1$.
Then Φ^* is equal to allocation $\hat{\Phi}_{\kappa_1}$. But this violates the assumption that Φ^* is different from any allocation $\hat{\Phi}_i$.

Induction assumption:
Assume that for $m = l - 1$ no deadlock-free allocation Φ^* exists with $\sum_{i \in \mathcal{N}_r} \Phi^*(B_{ir}) = K_r$ which satisfies (14).

Induction step: $m = l$
Consider the set of all cycles which contain the buffer sequence $(B_{\kappa_1 r}, B_{\kappa_2 r}, \ldots, B_{\kappa_{l-1} r}, B_{\kappa_l r'})$:

$$\begin{aligned} &C' = \\ &\{(B_{\kappa_1 r}, B_{\kappa_2 r}, \ldots, B_{\kappa_{l-1} r}, B_{\kappa_l r'}, B_{\kappa_{l+1} \rho_{l+1}}, B_{\kappa_{l+2} \rho_{l+2}}, \ldots, B_{\kappa_S \rho_S}) \mid \rho_s \in \{r, r'\}\} \end{aligned} \tag{18}$$

Assume a state where the jobs of chain r occupy all buffer space in $(B_{\kappa_1 r}, B_{\kappa_2 r}, \ldots, B_{\kappa_{l-1} r})$ and occupy servers $(\kappa_S, \kappa_2, \ldots, \kappa_{l-2})$. Assume further that jobs from chain r' occupy $B_{\kappa_l r'}$ and the server of station κ_{l-1}. Note that due to the induction assumption:

$$\sum_{s=1}^{l-1} \Phi^*(B_{\kappa_s r}) + (l-1) \leq K_r \tag{19}$$

and

$$\Phi^*(B_{l r'}) + 1 \leq K_{r'} \tag{20}$$

If the remaining jobs from class r and r' can be distributed over the remaining $S - l$ stations such that all these stations are blocked, then a deadlock occurs. Since any cycle $C \in \mathcal{C}'$ must be deadlock-free, the following condition must hold:

$$S - l > (K_r - \sum_{s=1}^{l-1} \Phi^*(B_{\kappa_s r}) - (l-1)) + (K_{r'} - B_{l r'} - 1) - z. \tag{21}$$

with $z = \sum_{s=l+1}^{S} \Phi^*(B_{\kappa_s \rho_s})$ $(\rho_s \in \{r, r'\})$. Let z_r $(z_{r'})$ be the sum of buffer capacities of chain r (r') which are allocated to buffers $B_{\kappa_{l+1}\rho}, B_{\kappa_{l+2}\rho}, \ldots, B_{\kappa_S \rho}$ of C, i.e.,

$$z_r = \sum_{s=l+1}^{S} (\Phi^*(B_{\kappa_s r}) \cdot \eta(C, \kappa_s, r)) \qquad z_{r'} = \sum_{s=l+1}^{S} (\Phi^*(B_{\kappa_s r'}) \cdot \eta(C, \kappa_s, r')) \tag{22}$$

with:

$$\eta(C, \kappa_s, r') = \begin{cases} 1 & \text{if } B_{\kappa_s r} \in C \\ 0 & \text{otherwise} \end{cases} \tag{23}$$

Note that $z_r, z_{r'} < l$ (equation (16)). Then for chain r:

$$\sum_{s=1}^{l} \Phi^*(B_{\kappa_s r}) + l - z_r = K_r. \tag{24}$$

We obtain for equation (21):

$$S - l > K_r - (K_r - l + z_r - B_{l r}) - (l-1) + K_{r'} - B_{l r'} - 1 - z, \tag{25}$$

We can choose a cycle C from $\mathcal{C}'$ such that $z_r + z \leq l$. Then :

$$S > K_{r'} + B_{l r'} - B_{l r}. \tag{26}$$

Now, consider the set of cycles

$$\begin{aligned} &\mathcal{C}'' = \\ &\{(B_{\kappa_1 r'}, B_{\kappa_2 r'}, \ldots, B_{\kappa_{l-1} r'}, B_{\kappa_l r}, B_{\kappa_{l+1}\rho_{l+1}}, B_{\kappa_{l+2}\rho_{l+2}}, \ldots, B_{\kappa_S \rho_S}) \mid \rho_s \in \{r, r'\}\} \end{aligned} \tag{27}$$

As before, we can construct cycles which have to satisfy the condition:

$$S > K_r + B_{l r} - B_{l r'}. \tag{28}$$

Adding up (26) and (28), we obtain:

$$2S > K_r + K_{r'} \tag{29}$$

But this is a contradiction, since $K_r \geq S$ and $K_{r'} \geq S$. Therefore, for $m = l$, Φ^* is not deadlock-free.

(c) $K_r \geq S$ and $\sum_{\rho \in \mathcal{R}\setminus\{r\}} K_\rho < S$
Since in this case

$$|S_r^{(C)}| > S - \sum_{\rho \in \mathcal{R}\setminus\{r\}} K_\rho \tag{30}$$

we can construct a tandem network $\overline{\Gamma}$ from Γ with $\overline{S}$ stations and $\overline{K}_\rho$ jobs in chain ρ with:

$$\begin{array}{lcll} \overline{S} & = & \sum_{\rho \in \mathcal{R}\setminus\{r\}} K_\rho & \\ \overline{\mathcal{R}} & = & \mathcal{R} & \\ \overline{K}_r & = & K_r - (S - \sum_{\rho \in \mathcal{R}\setminus\{r\}} K_\rho) & \\ \overline{K}_\rho & = & K_\rho & (\rho \in \mathcal{R} \setminus \{r\}) \end{array} \tag{31}$$

Per construction, Γ contains a deadlock if $\overline{\Gamma}$ contains a deadlock. Since $\sum_{\rho \in \mathcal{R}\setminus\{r\}} K_\rho \geq \overline{S}$, case (b) applies.

(d) $K_r < S$ and $\sum_{\rho \in \mathcal{R}\setminus\{r\}} K_\rho \geq S$
Since in this case

$$\sum_{\rho \in \mathcal{R}\setminus\{r\}} |S_\rho^{(C)}| > S - K_r \tag{32}$$

we can construct a tandem network $\overline{\Gamma}$ from Γ with $\overline{S}$ stations and $\overline{K}_\rho$ jobs in chain ρ with:

$$\begin{array}{lcll} \overline{S} & = & K_r & \\ \overline{\mathcal{R}} & = & \mathcal{R} & \\ \overline{K}_r & = & K_r & \\ \overline{K}_\rho & = & K_\rho - k_\rho & (\rho \in \mathcal{R} \setminus \{r\}) \end{array} \tag{33}$$

with:

$$\sum_{\rho \in \mathcal{R}\setminus\{r\}} k_\rho = S - K_r \tag{34}$$

Per construction, Γ contains a deadlock if $\overline{\Gamma}$ contains a deadlock. Since $K_r \geq \overline{S}$, case (b) applies. □

If $R = 1$, then $\sum_{\rho \in \mathcal{R}\setminus\{r\}} K_\rho = 0$. Therefore, an optimal buffer allocation for a single chain network can be obtained from Lemma 2 by:

$$\hat{\Phi}_i(B_j) = \begin{cases} K - S & \text{if } j = i \\ 0 & \text{if } j \neq i \end{cases} \tag{35}$$

From the proof of Lemma 2, we can follow immediately:

Corollary 1 *A buffer allocation Φ^* for a tandem network with S stations and R routing chains with $\sum_{\rho \in \mathcal{R}\setminus\{r\}} K_\rho \geq S$ (for $1 \leq r \leq R$) is optimal, if and only if there exists an $i \in \mathcal{N}$ such that for all $j \in \mathcal{N}$ and $r \in \mathcal{R}_j$:*

$$\Phi^*(B_{jr}) = \hat{\Phi}_i(B_{jr}) \tag{36}$$

3.2 Optimal Buffer Allocation for Networks with Tandem Sequences

With Lemma 2 we are able to directly give an optimal buffer allocation without running any optimization. However, the Lemma is only applicable for the special class of tandem networks. Next we show how to take advantage of the regularity of networks which contain a tandem sequence, i.e, contain a network like the network in Figure 2 as a subnetwork. The following Lemma prepares the so-called *Reduction Lemma* which will be presented at the end of this section. The *Reduction Lemma* allows to perform the allocation algorithm in [2] on a network where each tandem sequence is substituted by a single station. The optimal buffer allocation from the reduced network can be used to obtain an optimal allocation for the original network.

Lemma 3 *Given an optimal buffer allocation Φ^* for a network Γ. If Γ contains a tandem sequence of S stations $(\kappa_1, \kappa_2, \ldots, \kappa_{S-1}, \kappa_S)$, the following buffer allocation $\hat{\Phi}$ is optimal:*

$$\hat{\Phi}(B_{ir}) = \begin{cases} \sum_{v=1}^{S} \Phi^*(B_{\kappa_v r}) & \text{if } i = \kappa_1 \\ 0 & \text{if } i = \kappa_v \text{ and } 1 < v \leq S \\ \Phi^*(B_{ir}) & \text{otherwise} \end{cases} \quad \text{for } i \in \mathcal{N}, r \in \mathcal{R}_i. \tag{37}$$

Proof: $\hat{\Phi}$ is per definition minimal. It remains to be shown that $\hat{\Phi}$ yields a deadlock-free network. Assume the network with allocation $\hat{\Phi}$ is in a deadlock. The deadlock has to be along a cycle which includes one buffer from each of stations $(\kappa_1, \kappa_2, \ldots, \kappa_{S-1}, \kappa_S)$. Otherwise, Γ has a deadlock under Φ^*. Note that Φ^* is equal to $\hat{\Phi}$ for buffers of stations not in the tandem sequence. Let the buffer cycle which contains a possible deadlock be given by:

$$\widehat{C} = (B_{\sigma_1\rho_1}, B_{\sigma_2\rho_2}, \ldots, B_{\kappa_1\tau_1}, B_{\kappa_2\tau_2}, \ldots, B_{\kappa_{S-1}\tau_{S-1}}, B_{\kappa_S\tau_S}, \ldots, B_{\sigma_{t-1}\rho_{t-1}}, B_{\sigma_t\rho_t}) \tag{38}$$

Consider the tandem network Ψ obtained from Γ by eliminating the stations not belonging to the tandem sequence. Let $\mathcal{N}_\Psi$, $\mathcal{R}_\Psi$, $K_{\Psi,r}$ denote the set of stations, the set of classes and the number of jobs, respectively, in Ψ, with:

$$\begin{aligned} \mathcal{N}_\Psi &= \{B_{\kappa_1\tau_1}, B_{\kappa_2\tau_2}, \ldots, B_{\kappa_{S-1}\tau_{S-1}}, \ldots, B_{\sigma_S\rho_S}\}, \\ \mathcal{R}_\Psi &= \mathcal{R}_{\kappa_1}, \\ K_{\Psi,r} &= K_r - \Big(\sum_{1<u\leq t \wedge \sigma_u \in \overline{S}_r^{(\widehat{C})}} \Phi^*(B_{\sigma_u r}) + |\overline{S}_r^{(\widehat{C})}| \Big), \end{aligned} \tag{39}$$

where:

$$\overline{S}_r^{(C)} = S_r^{(C)} \setminus \{\kappa_1, \kappa_2, \ldots, \kappa_{S-1}, \kappa_S\}. \tag{40}$$

Any allocation Φ for network Γ defines an allocation for Ψ. Per definition of Ψ, allocation $\hat{\Phi}$ for Ψ violates the DLF condition for cycle $\tilde{C}$ with:

$$\tilde{C} = (B_{\kappa_1\tau_1}, B_{\kappa_2\tau_2}, \ldots, B_{\kappa_{S-1}\tau_{S-1}}, B_{\sigma_S\rho_S}), \tag{41}$$

if $\hat{C}$ in Γ under $\hat{\Phi}$ violates the DLF condition. According to Lemma 2, we can find a minimal and deadlock-free buffer allocation for Ψ with:

$$\hat{\Phi}(B_{i\tau}) = 0, \qquad \text{if } i = \kappa_v \text{ and } 1 < v \leq S. \tag{42}$$

Therefore, if $\hat{\Phi}$ contains a deadlock, then the assumption that Φ^* is optimal for Γ does not hold. □

Lemma 4 (Reduction Lemma) *Given a network Γ with a tandem sequence $(\kappa_1, \kappa_2, \ldots, \kappa_{S-1}, \kappa_S)$ with $r \in \mathcal{R}_{\kappa_1}$. Obtain a network Γ' by substituting the tandem sequence in Γ by a single station $\tilde{\kappa}$. Let $\tilde{\Phi}$ be a minimal allocation for Γ'. Define for all $r \in \mathcal{R}_{\tilde{\kappa}}$*

$$M_{\tilde{\kappa}r} = \max\{\sum_{\rho \in \mathcal{R}\setminus\{r\}} (K_\rho - \sum_{1<u\leq t \wedge \sigma_u \in S_\rho^{(C)}} \tilde{\Phi}(B_{\sigma_u\rho}) - |S_\rho^{(C)}|) \mid C \text{ Cycle in } \Gamma',\ B_{\tilde{\kappa}r} \in C\}. \tag{43}$$

If the condition holds that:

$$M_{\tilde{\kappa}r} + \tilde{\Phi}(B_{\tilde{\kappa}r}) \geq S \quad \text{for all } r \in \mathcal{R}_{\tilde{\kappa}}, \tag{44}$$

then the following buffer allocation Φ^* is optimal for Γ:

$$\Phi^*(B_{ir}) = \begin{cases} \tilde{\Phi}(B_{\tilde{\kappa}r}) & \text{if } i = \kappa_1 \text{ and } M_{\tilde{\kappa}r} \geq S-1 \\ \tilde{\Phi}(B_{\tilde{\kappa}r}) - (S-1) + M_{\tilde{\kappa}r} & \text{if } i = \kappa_1 \text{ and } M_{\tilde{\kappa}r} < S-1 \\ 0 & \text{if } i = \kappa_v \text{ and } 1 < v \leq S \\ \tilde{\Phi}(B_{ir}) & \text{otherwise} \end{cases} \tag{45}$$
$$\text{for } i \in \mathcal{N}, r \in \mathcal{R}_i.$$

Proof:

(1) Φ^* is a deadlock-free allocation

We only have to consider cycles which include buffers from stations of the tandem sequence. Assume a cycle C' in Γ' which includes buffer $B_{\tilde{\kappa}\tau_1}$ ($\tau_1 \in \mathcal{R}_{\tilde{\kappa}}$):

$$C' = (B_{\sigma_1\rho_1}, B_{\sigma_2\rho_2}, \ldots, B_{\tilde{\kappa}\tau_1}, \ldots, B_{\sigma_{t-1}\rho_{t-1}}, B_{\sigma_t\rho_t}) \tag{46}$$

C' corresponds to a set of cycles in Γ:

$$\mathcal{C}'' = \{(B_{\sigma_1\rho_1}, B_{\sigma_2\rho_2}, \ldots, B_{\kappa_1\tau_1}, B_{\kappa_2\tau_2}, \ldots, B_{\kappa_S\tau_S}, \ldots, B_{\sigma_{t-1}\rho_{t-1}}, B_{\sigma_t\rho_t}) \mid \tau_v \in \mathcal{R}_{\tau_1}, 1 \leq v \leq S\} \tag{47}$$

(a) $M_{\tilde{\kappa}r} \geq S-1$

Then for all $C'' \in \mathcal{C}''$ and for all $r \in \mathcal{R}$ the following holds:

$$\sum_{1<u\leq t \wedge \sigma_u \in S_r^{(C')}} \Phi^*(B_{\sigma_u r}) = \sum_{1<u\leq t \wedge \sigma_u \in S_r^{(C'')}} \tilde{\Phi}(B_{\sigma_u r}), \tag{48}$$

and:

$$|S_r^{(C')}| \leq |S_r^{(C'')}|, \tag{49}$$

Since C' is deadlock-free in Γ' with $\tilde{\Phi}$, it follows from (48) and (49) that all $C'' \in \mathcal{C}''$ are deadlock-free in Γ with Φ^*.

(b) $M_{\tilde{\kappa}r} < S-1$
Recall from (45) that in this case $\Phi^*(B_{\tilde{\kappa}r}) < \tilde{\Phi}(B_{\tilde{\kappa}r})$. If the DLF condition for cycle C' in Γ' is satisfied by chain $\rho \neq \tau_1$, then equations (48) and (49) hold for chain ρ. But then, the DLF condition for $C'' \in \mathcal{C}''$ in Γ is satisfied by chain $\rho \neq \tau_1$. Now assume that chain τ_1 satisfies the DLF condition for cycle C' in Γ', i.e.,

$$\sum_{1<u\leq t\wedge\sigma_u\in S_{\tau_1}^{(C')}} \Phi^*(B_{\sigma_u\tau_1}) + |S_{\tau_1}^{(C')}| > K_{\tau_1} \tag{50}$$

Since $M_{\tilde{\kappa}r} < S-1$, at least $((S-1)-M_{\tilde{\kappa}r})$ of the servers of stations $(\kappa_2, \kappa_3, \ldots, \kappa_S)$ have to be occupied by jobs of chain τ_1 if a deadlock persists in a cycle $C'' \in \mathcal{C}''$. But then:

$$|S_{\tau_1}^{(C'')}| \geq |S_{\tau_1}^{(C')}| + ((S-1)-M_{\tilde{\kappa}\tau_1}) \tag{51}$$

Note that from (45):

$$\sum_{1<u\leq t\wedge\sigma_u\in S_{\tau_1}^{(C')}} \Phi^*(B_{\sigma_u\tau_1}) = \sum_{1<u\leq t\wedge\sigma_u\in S_{\tau_1}^{(C'')}} \tilde{\Phi}(B_{\sigma_u\tau_1}) - ((S-1)-M_{\tilde{\kappa}\tau_1}). \tag{52}$$

From equations (51) and (52) we see immediately, that chain τ_1 satisfies the DLF condition for all $C'' \in \mathcal{C}''$ of Γ.

(2) Φ^* is minimal
Because of Lemma 3 we only need to consider allocations which assign capacities to station κ_1 and no capacities at all to stations κ_v $(1 < v \leq S)$ in Γ. Since $\tilde{\Phi}$ is optimal for Γ', network Γ' will have a deadlock if we assign less capacity than $\tilde{\Phi}(B_{\tilde{\kappa}\tau_1})$ to buffer $B_{\tilde{\kappa}\tau_1}$. Take a cycle $\widehat{C}$ in Γ' with, which satisfies the DLF condition for chain $\tilde{\kappa}$ if a capacity of $\tilde{\Phi}(B_{\tilde{\kappa}\tau_1})$ is assigned to $B_{\tilde{\kappa}\tau_1}$, but does not so if the capacity is $(\tilde{\Phi}(B_{\tilde{\kappa}\tau_1})-1)$. $\widehat{C}$ has the same structure as the cycle in (46). Let $\widehat{\mathcal{C}}$ be the set of cycles in Γ which correspond to $\widehat{C}$ in Γ'. $\widehat{\mathcal{C}}$ has the same structure as set C' in (47). For a cycle $\widehat{\widehat{C}} \in \widehat{\mathcal{C}}$ we distinguish two cases:

(a) $M_{\tilde{\kappa}r} \geq S-1$
In this case, the servers of stations $(\kappa_2, \kappa_3, \ldots, \kappa_S)$ can be occupied by jobs of chains from $\mathcal{R}\setminus\{r\}$ such that no $\rho \in \mathcal{R}\setminus\{r\}$ satisfies the DLF condition for cycle $\widehat{\widehat{C}}$. Then:

$$\sum_{1<u\leq t\wedge\sigma_u\in S_{\tau_1}^{(\widehat{\widehat{C}})}} \Phi^*(B_{\sigma_u\tau_1}) = \sum_{1<u\leq t\wedge\sigma_u\in S_{\tau_1}^{(\widehat{C})}} \tilde{\Phi}(B_{\sigma_u\tau_1}), \tag{53}$$

and:

$$|S_{\tau_1}^{(\widehat{\widehat{C}})}| = |S_{\tau_1}^{(\widehat{C})}|, \tag{54}$$

Therefore, any allocation with:

$$\Phi^*(B_{\kappa_1\tau_1}) < \tilde{\Phi}(B_{\tilde{\kappa}\tau_1}) \tag{55}$$

does not satisfy the DLF condition for chain τ_1, and thus, a deadlock persists for all $\hat{\hat{C}} \in \hat{\hat{\mathcal{C}}}$.

(b) $M_{\tilde{\kappa}\tau} < S-1$

Here $M_{\tilde{\kappa}\tau_1}$ of the servers of stations $(\kappa_2, \kappa_3, \ldots, \kappa_S)$ can be occupied by jobs of chains from $\mathcal{R} \setminus \{r\}$ without satisfying the DLF condition for any chain $\rho \in \mathcal{R} \setminus \{r\}$. If condition (44) holds, then the remaining $((S-1) - M_{\tilde{\kappa}\tau_1})$ servers can be occupied by jobs from chain τ_1. Then:

$$\begin{aligned}
&\sum_{1<u\le t\wedge\sigma_u\in S_{\tau_1}^{(\hat{\hat{C}})}\setminus\{\kappa_1\}} \Phi^*(B_{\sigma_u\tau_1}) + \Phi^*(B_{\kappa_1\tau_1}) + |S_{\tau_1}^{(\hat{\hat{C}})}| \\
&= \sum_{1<u\le t\wedge\sigma_u\in S_{\tau_1}^{(\hat{C})}\setminus\{\kappa_1\}} \Phi^*(B_{\sigma_u\tau_1}) + (\tilde{\Phi}(B_{\tilde{\kappa}\tau_1}) - (S-1) + M_{\tilde{\kappa}\tau_1}) + (|S_{\tau_1}^{(\hat{C})}| + (S-1) - M_{\tilde{\kappa}\tau_1}) \\
&= \sum_{1<u\le t\wedge\sigma_u\in S_{\tau_1}^{(\hat{C})}\setminus\{\tilde{\kappa}\}} \tilde{\Phi}(B_{\sigma_u\tau_1}) + \tilde{\Phi}(B_{\tilde{\kappa}\tau_1}) + |S_{\tau_1}^{(\hat{C})}|
\end{aligned} \tag{56}$$

But then, any allocation with:

$$\Phi^*(B_{\kappa_1\tau_1}) < \tilde{\Phi}(B_{\tilde{\kappa}\tau_1}) - (S-1) + M_{\tilde{\kappa}\tau_1} \tag{57}$$

does not satisfy the DLF condition for chain τ_1. We follow from the construction of $\hat{\hat{C}}$ that network Γ contains a deadlock for all $\hat{\hat{C}} \in \hat{\hat{\mathcal{C}}}$. □

4 Example

In Figure 3 we show a multiple chain queueing network with a tandem sequence of length $S = 5$. Obtaining an optimal buffer allocation algorithm for networks which include a tandem sequence is a formidable task because of the high number of cycles. Even after eliminating those cycles that are a superset of other cycles according to Lemma 1, the network in Figure 3 contains 2561 cycles that have to be considered for the buffer allocation algorithm. With the *Reduction Lemma* (Lemma 4) we can replace the tandem sequence, i.e., stations (6, 7, 8, 9, 10), by a single station. The reduced network contains only 11 cycles which have to be considered for the optimization. Since the *Reduction Lemma* is valid only if the condition in equation (44) is satisfied, we have to show that the reduction is valid. Note that none of the cycles $(C_1, C_2, \ldots, C_{11})$ includes buffers from all routing chains and $K_r > S$ for $r = 1, 2, 3, 4$. Therefore, condition (44) is clearly satisfied for all routing chains.

Applying the optimization algorithm in [2] to the reduced network yields an optimal buffer allocation $\tilde{\Phi}$. If we let $B_{\{6,7,8,9,10\},r}$ (for $r = 1, 2, 3, 4$) denote the chain-r-buffer of the station which replaces the tandem sequence in the reduced network we obtain an

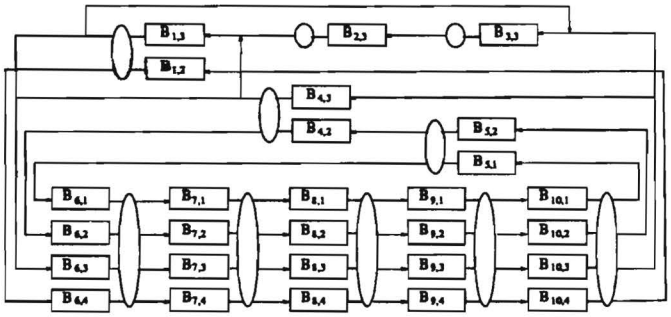


Figure 3: Multiple Chain Queueing Network with a Tandem Sequence.

optimal allocation for the network in Figure 3 by:

$$\Phi^*(B_{ir}) = \begin{cases} \tilde{\Phi}(B_{\{6,7,8,9,10\},r}) & \text{if } i = 6 \\ 0 & \text{if } i \in \{7,8,9,10\} \\ \tilde{\Phi}(B_{ir}) & \text{otherwise} \end{cases}$$

5 Conclusions

We investigated deadlock-free buffer allocations in blocking network with multiple routing chains. In order to show that a network is deadlock-free, all cycles in the network need to satisfy the deadlock freedom condition. Networks with so-called tandem sequences are characterized by a high number of cycles. Therefore, finding all cycles becomes computational impractical even for small networks. We showed that the problem of finding all cycles can be dealt with by substituting tandem sequences of a queueing network with single stations. The so-called *Reduction Lemma* guarantees that an optimal deadlock-free buffer allocation for the substitute network is an optimal deadlock-free buffer allocation for the original network. In addition, an optimal deadlock-free buffer allocation for networks consisting of only one tandem sequence can be directly obtained without applying any optimization procedure.

References

[1] I. F. Akyildiz and H. G. Perros, "Queueing Networks with Finite Capacity Queues", *Special Issue in Performance Evaluation*, Vol. 10, No. 4, December 1989.

[2] I. F. Akyildiz and J. Liebeherr, "Optimal Deadlock Free Buffer Allocation in Multiple Chain Blocking Networks of Queues", *Proc. IFIP TC7/WG 7.3 International Conference on Performance of Distributed Systems and Integrated Communication Networks*, Kyoto, Japan, September 1991, pp. 217-236.

[3] S. Kundu and I. F. Akyildiz, "Deadlock Free Buffer Allocation in Closed Queueing Networks", *Queueing Systems*, Vol. 4, No. 1, January 1989, pp. 47 - 56.

Queueing Networks with Finite Capacity
R.O. Onvural and I.F. Akyildiz (Editors)

EXACT ANALYSIS OF MULTI-JOB CLASS NETWORKS OF QUEUES WITH BLOCKING-AFTER-SERVICE*

Ian F. Akyildiz and Chueng-Chiu Huang

School of Electrical Engineering

Georgia Institute of Technology

Atlanta, GA 30332

U.S.A.

Abstract

Two models of closed queueing networks with blocking after service and multiple job classes are analyzed. The first model is a network with N stations and each station has either Type II or Type III. The second model is a star like queueing network also called as central server model in which the stations may have either Type I or Type IV, with the condition that the neighbors of these stations must be of Type II or Type III such that blocking will be caused only by this set of station types. Exact product form solutions are obtained for the equilibrium state probabilities in both models. Formulae for performance measures such as throughput and mean number of jobs are also derived.

Key Words: Performance Evaluation, Queueing Networks, Finite Buffer Capacity, Blocking, Deadlock, Equilibrium State Probabilities, Throughput, Mean Number of Jobs.

* **This work was supported by National Science Foundation (NSF) under Grant No. CCR-90-11981.**

1 Introduction

In recent years there has been an increased interest in the analysis of queueing networks with blocking. This is probably due to the realization that these queueing networks are useful in modelling computer systems, communication networks, and flexible manufacturing systems. The set of rules that dictate when a station becomes blocked and when it becomes unblocked is commonly referred to as the *blocking* mechanism. There are basically only a few blocking mechanisms that have been extensively studied in the literature [3,13]. We consider the so-called *blocking after service* [3,13] (in short form BAS) mechanism, i.e., if a job finishes service at a station and wants to enter a station which is full, it stays in the server of the source station, waiting for a space to be available in the destination station. This blocking policy is also known type 1 blocking, transfer blocking, manufacturing blocking, production blocking and non-immediate blocking in the literature [3,13]. Several papers consider this blocking policy [1,2,3,4,5,6,7,9,12,13,14]. Onvural and Perros [14] and Akyildiz and von Brand [4] consider closed queueing networks with BAS mechanism for limited number of jobs. Akyildiz and Liebeherr [5] determine necessary and sufficient conditions for deadlock-freedom. Akyildiz and von Brand [4] considered queueing networks with BAS mechanism. In the first model they analyze a two-station model with different types of stations and single class of jobs. They derive an exact product form solution for the equilibrium state probabilities. In [4] they also analyze a second model with N stations, multiple classes of jobs and the limited number of jobs $K = \min_{i \in N} \{B_i\} + 1$, where B_i is the buffer capacity of station i. If the stations that will cause blocking are type I station in $BCMP$ model, an exact product form solution is obtained.

In this paper we consider two models of closed queueing networks with BAS mechanism. In both models we assume that there are N stations and R classes of jobs. The first model is a general topology network with N stations where each station is either of Type II or Type III [8,10,11]. The second model is a star like queueing network (also known as a central server model) where the central server must have the station Type I or Type IV [8,10,11] and the neighbors must be either of Type II or Type III stations. In the second model we assume that the blocking will be caused only by the central server. For both models we obtain exact product form solution for equilibrium state probabilities. We also derive formulae for performance measures such as throughput and mean number of jobs.

The paper is organized as follows: In section 2 introduce common notations and definitions for both models. In section 3 we analyze the first model. In section 4 we investigate the central server model. In section 5 we conclude the paper.

2 Basic Model Description

Both models contain N stations, R job classes and K total number of jobs. The types of stations we consider in this paper are from $BCMP$ model [8,10,11]:

- Type I: The service discipline is first-come-first-served (FCFS); all job classes have the same service time distribution and the service rate can be state-dependent where $\mu(k)$ will denote the service rate with k jobs.
- Type II: There is a single server and the service discipline is processor sharing (PS). Each job class may have a distinct service time distribution.
- Type III: The number of servers is greater or equal to the maximum number of jobs which can be queued at the station; infinite servers (IS). Each job class may have a distinct service time distribution.
- Type IV: There is a single server and the queueing discipline is last-come-first-served preemptive-resume (LCFS-PR). Each job may have a distinct service time distribution.

Here we assume that all service times follow exponential distribution which depend on the station and on the class. Note that our results can easily be extended to general service time distributions for Types II, III and IV. For the sake of simplicity here we give results only for exponential cases.

The transition probabilities are denoted by $p_{ir,js}$, $1 \leq i,j \leq N$ and $1 \leq r,s \leq R$, where a class r job departs from station i and visits station j and becomes a class s job. We also assume that the routing matrix $\mathbf{P} = [p_{i,r,j,s}]$, is irreducible. We define

$$\alpha_{js} = \sum_{i=1}^{N} \sum_{r=1}^{R} \alpha_{ir} p_{ir,js}. \tag{1}$$

Let the vector $\mathbf{n} = (\mathbf{x}_1, \cdots, \mathbf{x}_N)$ denote the number of jobs in N stations and their positions in each station, i.e., $\mathbf{x}_i = (x_{i1}, \cdots, x_{in_i})$, where x_{il} denotes the class of the job in station i at the position l. Let n_i be the number of jobs in station i, and $n_i = n_{i1} + \cdots + n_{iR}$, where n_{ir} is the number of class r jobs in station i. We note that if station i is of Type II or III, then there is no requirement for the order of the jobs, thus, we denote $\mathbf{x}_i$ as $\mathbf{n}_i$, $\mathbf{n}_i = (n_{i1}, \cdots, n_{iR})$.

A product form solution for equilibrium state probabilities exists when there is no blocking in the network [8,10,11]:

$$p(\mathbf{n}) = C \prod_{i=1}^{N} f_i(\mathbf{x}_i), \tag{2}$$

where C is the normalization constant such that the sum of equilibrium state probabilities will be equal to one and $f_i(\mathbf{x}_i)$ is defined by the type of station i,

$$f_i(\mathbf{x}_i) = \begin{cases} \left(\frac{1}{\mu_i}\right)^{n_i} \prod_{l=1}^{n_i} \alpha_{ix_{il}}, & \text{if station } i \text{ is Type } I \\ n_i! \prod_{r=1}^{R} \frac{1}{n_{ir}!} \left(\frac{\alpha_{ir}}{\mu_{ir}}\right)^{n_{ir}}, & \text{if station } i \text{ is Type } II \\ \prod_{r=1}^{R} \frac{1}{n_{ir}!} \left(\frac{\alpha_{ir}}{\mu_{ir}}\right)^{n_{ir}}, & \text{if station } i \text{ is Type } III \\ \prod_{l=1}^{n_i} \left(\frac{\alpha_{ix_{il}}}{\mu_{ix_{il}}}\right). & \text{if station } i \text{ is Type } IV \end{cases} \tag{3}$$

We define T_1 as the set of Type I or IV stations, and T_2 as the set of Type II or III stations. Since we are considering the models with blocking, we need to define B_i as the capacity of station i (i.e., buffer and server capacity). In addition to avoid the self loop that cause *deadlock* we assume that

$$p_{ii} = 0, \text{ for } i = 1, \cdots, N.$$

For convenience we define the following quantities for each station type. For type II and III station, the service rate are dependent on the number of jobs in the station and on their classes but independent of their order. Thus, we let $\mathbf{n}_i = (n_{i1}, \cdots, n_{iR})$ for $i \in T_2$, where n_{ir} denotes the number of class r active jobs in station i. A job is said to be active if it is not blocked. Also we let $n_i = n_{i1} + \cdots + n_{iR}$ be the total number of active jobs in station i for $i \in T_2$.

Let b denote the number of blocked jobs. Thus, with the active jobs we have

$$K = \sum_{i=1}^{N} n_i + b, \tag{4}$$

with

$$0 \leq b \leq K - \min_{1 \leq i \leq N} \{B_i\}. \tag{5}$$

We assume that a job will choose its destination station and the class when it finishes its service. Considering the b blocked jobs we need to know their locations and their classes. We also assume that the blocked jobs will join that station, which caused blocking according to $First - Come - First - In$ $(FCFI)$ scheduling discipline. Thus, we let $\mathbf{y} = (\mathbf{y}_i)$ and $\mathbf{y}_i = (y_{i1}, \cdots, y_{ib_i})$ if $b_i \geq 1$ otherwise $\mathbf{y}_i = \emptyset$ where b_i denotes the number of jobs blocked by station i, and y_{il}, $1 \leq y_{il} \leq N$ denotes the location of the l^{th} job blocked by station i. Similarly, $\mathbf{z} = (\mathbf{z}_i)$ and $\mathbf{z}_i = (z_{i1}, \cdots, z_{ib_i})$ where z_{il}, $1 \leq z_{il} \leq R$ denotes the class of the lth blocked job. To have the complete information about the system we define the state space $S = \{(\mathbf{n}, \mathbf{x}, \mathbf{y}, \mathbf{z})\}$, where $\mathbf{x} = (\mathbf{x}_i, i \in T_1)$ and $\mathbf{n} = (\mathbf{n}_i, i \in T_2)$. For a given a state $(\mathbf{n}, \mathbf{x}, \mathbf{y}, \mathbf{z})$ the service rate of station i for $i \in T_2$ is $\mu_i(\mathbf{n}_i) = \mu_{i1}(n_{i1}) + \cdots + \mu_{iR}(n_{iR})$ where $\mu_{ir}(n_{ir}) = n_{ir}\mu_{ir}$ if station i is of Type III, and if station i is of Type II, then $\mu_{ir}(n_{ir}) = \frac{n_{ir}}{n_i}\mu_{ir}$. For $i \in T_1$ then the service rate is: if station i is of type I, then $\mu_i(\mathbf{x}_i) = \mu_i$ for $n_i > 0$, otherwise it is equal to 0. If station i is of Type IV, then $\mu_i(\mathbf{x}_i) = \mu_{in_i}$ for $n_i \leq B_i$.

Let e_{ir} denote an NxR matrix with all zero elements except the element at the position of ith row and rth column with value 1.

3 General Topology Network with Type II and III Stations

In this section we consider a general topology network with only Type II and III stations. We assume the following constraint in the model:

$$K < \min_{1 \le i,j \le N; p_{ij} > 0} \{B_i + B_j\}, \tag{6}$$

This constraint says that only one of the N stations may cause blocking at any time. The reason of making this assumption is to avoid more than two jobs moving at one event. Thus, the state space is reduced to $S = (\mathbf{n}, \mathbf{y}, \mathbf{z})$, where $\mathbf{y}$ and $\mathbf{z}$ are b by 1 vectors, because if there are any blocked jobs, then they are all blocked by the same station.

Now we consider the transitions of the jobs for a nonblocking state $(\mathbf{n}, \mathbf{y}, \mathbf{z})$, where $\mathbf{y}$ and $\mathbf{z}$ are empty vectors $\emptyset$. In this case the system will reach the state $(\mathbf{n} - e_{ir} + e_{js}, \emptyset, \emptyset)$ with rate $\mu_{ir}(n_{ir}) p_{ir,js}$, if $n_j \neq B_j$. Otherwise, a class r job from station i will be blocked by station j and the transition will occur from state $(\mathbf{n}, \emptyset, \emptyset)$ to state $\left(\mathbf{n} - e_{ir}, \mathbf{y}', , \mathbf{z}'\right)$ where $\mathbf{y}' = (y_1)$ and $y_1 = i$. The same occurs for $\mathbf{z}' = (z_1)$ and $z_1 = s$. The rate to reach this state is the same as before, i.e., $\mu_{ir}(n_{ir}) p_{ir,js}$.

Now we consider the transitions from a blocking state. Suppose station k is full, i.e., $n_k = B_k$ and the current state is $(\mathbf{n}, \mathbf{y}, \mathbf{z})$, for $b \geq 0$. Then a class r job leaves station i and moves to station j, for $i \neq k, j \neq k$ and becomes a class s job and the system reaches the state $(\mathbf{n} - e_{ir} + e_{js}, \mathbf{y}, \mathbf{z})$ with rate $\mu_{ir}(n_{ir}) p_{ir,js}$. If a class r job moves from station i to station k and becomes a class s job, then the state $(\mathbf{n} - e_{ir}, \mathbf{y} + y_{b+1}, \mathbf{z} + z_{b+1})$ and $y_{b+1} = i$, $z_{b+1} = s$ will be reached. Note that in this case the transition rate is $\mu_{ir}(n_{ir}) p_{ir,js}$.

Let $\mathbf{y} = (y_1, \cdots, y_b)$ then the operator $\mathbf{y} + y_{b+1}$ will be $(y_1, \cdots, y_{b+1})$. Similar is also valid for the operator $\mathbf{z} + z_{b+1}$. When considering a class r job which moves from station k to station j and becomes a class s job with the transition rate $\mu_{kr}(n_{kr}) p_{kr,js}$, then one of the first blocked jobs will move to station k at the same time. Thus, the new state is $\left(\mathbf{n} - e_{kr} + e_{js} + e_{kz}, \mathbf{y}', \mathbf{z}'\right)$, with $\mathbf{y}' = \left(y_1', \cdots y_{b-1}'\right)$, $y_l' = y_{l+1}$ and $\mathbf{z}' = \left(z_1', \cdots z_{b-1}'\right)$, $z_l' = z_{l+1}$ for $1 \leq l \leq b-1$.

From these state transitions we obtain the global balance equation for a state $(\mathbf{n}, \mathbf{y}, \mathbf{z}) \in S$ if $n_i < B_i$ for $1 \leq i \leq N$, thus $\mathbf{y} = \emptyset$ and $\mathbf{z} = \emptyset$:

$$p(\mathbf{n}, \emptyset, \emptyset) \sum_{i=1}^{N} \sum_{r=1}^{R} \mu_{ir}(n_{ir}) = \sum_{i=1}^{N} \sum_{r=1}^{R} \sum_{j=1}^{N} \sum_{s=1}^{R} p(\mathbf{n} + e_{ir} - e_{js}, \emptyset, \emptyset) \mu_{ir}(n_{ir} + 1) p_{ir,js}, \tag{7}$$

and the global balance equation for a blocking state (suppose $n_k = B_k$):

$$
\begin{aligned}
& p(\mathbf{n},\mathbf{y},\mathbf{z}) \sum_{i=1}^{N}\sum_{r=1}^{R} \mu_{ir}(n_{ir}) \\
&= \sum_{i=1,i\neq k}^{N}\sum_{r=1}^{R}\sum_{j=1,j\neq k}^{N}\sum_{s=1}^{R} p(\mathbf{n}+e_{ir}-e_{js},\mathbf{y},\mathbf{z})\,\mu_{ir}(n_{ir}+1)\,p_{ir,js} \\
&+ \sum_{j\neq k}\sum_{s=1}^{R}\sum_{r=1}^{R}\sum_{y_0=1}^{N}\sum_{z_0=1}^{R} p(\mathbf{n}+e_{kr}-e_{js}-e_{kz_0},\mathbf{y}+y_0,\mathbf{z}+z_0)\,\mu_{kr}(n_{kr})\,p_{kr,js} \\
&+ \sum_{s=1}^{R} p(\mathbf{n}+e_{y_b s},\mathbf{y}-y_b,\mathbf{z}-z_b)\,\mu_{y_b s}(n_{y_b s}+1)\,p_{y_b s,kz_b}. \qquad (8)
\end{aligned}
$$

The first term on the right hand side of (8) is the total flow-in rate to the state $(\mathbf{n},\mathbf{y},\mathbf{z})$ from state $(\mathbf{n}+e_{ir}-e_{js},\mathbf{y},\mathbf{z})$. We note that the moving job is from station i to station j, $i\neq k, j\neq k$. The second term on the right-hand side of (8) is the flow-in rate to state $(\mathbf{n},\mathbf{y},\mathbf{z})$ by a class r job moving out from station k, and in the mean time another blocked job, which is class z_0, will move into station k from station y_0. As a consequence a transition occurs from the state $(\mathbf{n}+e_{kr}-e_{js}-e_{kz_0},\mathbf{y}+y_0,\mathbf{z}+z_0)$ to the state $(\mathbf{n},\mathbf{y},\mathbf{z})$.

For notational convenience we let y_0, z_0 denote the location and the class of the blocked job that will move into station k while a job leaves station k and joins station i. The third term in (8) is the flow-in rate to the state $(\mathbf{n},\mathbf{y},\mathbf{z})$ by a joining job at station k.

We note that a given state will be visited and the departure job is from a station i, $i\neq k$, then we know where the job comes from, i.e., $y_b=i$ and its class is switched from s to z_b. So that the transition is from the state $(\mathbf{n}+e_{y_b s},\mathbf{y}-y_b,\mathbf{z}-z_b)$ to the state $(\mathbf{n},\mathbf{y},\mathbf{z})$.

Theorem 1. The model has the following product form solution for equilibrium state probabilities

$$
p(\mathbf{n},\mathbf{y},\mathbf{z}) = C\prod_{i=1}^{N} f_i(\mathbf{n}_i)\left(\prod_{l=1}^{b}\frac{\sum_{t=1}^{R}\alpha_{y_l t}p_{y_l t,iz_l}}{\sum_{t=1}^{R}\mu_{it}(n_{it})}\right)^{I_{\{\mathbf{n}_i=B_i\}}} \qquad (9)
$$

where $f_i(\mathbf{n}_i)$ is defined

$$
f_i(\mathbf{n}_i) = \left\{ \begin{array}{ll} n_i!\prod_{r=1}^{R}\frac{1}{n_{ir}!}\left(\frac{\alpha_{ir}}{\mu_{ir}}\right)^{n_{ir}} & \text{if station } i \text{ is type } II \\ \prod_{r=1}^{R}\frac{1}{n_{ir}!}\left(\frac{\alpha_{ir}}{\mu_{ir}}\right)^{n_{ir}} & \text{if station } i \text{ is type } III \end{array} \right). \qquad (10)
$$

and C is a normalization constant

$$
C^{-1} = \sum_{(\mathbf{n},\mathbf{y},\mathbf{z})\in S}\prod_{i=1}^{N} f_i(\mathbf{n}_i)\left(\prod_{l=1}^{b}\frac{\sum_{t=1}^{R}\alpha_{y_l t}p_{y_l t,iz_l}}{\sum_{t=1}^{R}\mu_{it}(n_{it})}\right)^{I_{\{\mathbf{n}_i=B_i\}}} \qquad (11)
$$

Proof. We escape the proof of the nonblocking states since it is exactly the same as in the *BCMP* model [8,10,11]. For the global balance equation of blocking states we assume that the

state k has $n_k = B_k$ and the number of blocked jobs is b, then for (9) and (10) we have

$$p(\mathbf{n} + e_{ir} - e_{js}, \mathbf{y}, \mathbf{z}) = p(\mathbf{n}, \mathbf{y}, \mathbf{z}) \frac{\alpha_{ir}}{\mu_{ir}(n_{ir}+1)} \frac{\mu_{js}(n_{js})}{\alpha_{js}} \tag{12}$$

if $i \neq k$ and $j \neq k$.

Next we consider the departure event of a job leaving station $i \neq k$ and visiting station k

$$\begin{aligned} & p(\mathbf{n} + e_{kr} - e_{js} - e_{kz_0}, \mathbf{y} + y_0, \mathbf{z} + z_0) \\ & = p(\mathbf{n}, \mathbf{y}, \mathbf{z}) \frac{\alpha_{kr}}{\mu_{kr}(n_{kr}+1)} \frac{\mu_{js}(n_{js})}{\alpha_{js}} \frac{\mu_{kz_0}(n_{kz_0})}{\alpha_{kz_0}} \frac{\sum_{t=1}^{R} \alpha_{y_0 t} p_{y_0 t, kz_0}}{\sum_{t=1}^{R} \mu_{kt}(n_{kt})} \end{aligned} \tag{13}$$

and for the departure event of a job leaving station k and visiting station i

$$p(\mathbf{n} + e_{y_b s}, \mathbf{y} - y_b, \mathbf{z} - z_b) = p(\mathbf{n}, \mathbf{y}, \mathbf{z}) \frac{\alpha_{y_b s}}{\mu_{y_b s}(n_{y_b s}+1)} \frac{\sum_{t=1}^{R} \mu_{kt}(n_{kt})}{\sum_{t=1}^{R} \alpha_{y_b t} p_{y_b t, kz_b}}. \tag{14}$$

By substituting (12) – (14) to the right-hand side of (8) separately and using (1), we obtain

$$\begin{aligned} & \sum_{i=1, i\neq k}^{N} \sum_{r=1}^{R} \sum_{j=1, j\neq k}^{N} \sum_{s=1}^{R} p(\mathbf{n} + e_{ir} - e_{js}, \mathbf{y}, \mathbf{z}) \mu_{ir}(n_{ir}+1) p_{ir,js} \\ & = p(\mathbf{n}, \mathbf{y}, \mathbf{z}) \sum_{j=1, j\neq k}^{N} \sum_{s=1}^{R} \frac{\mu_{js}(n_{js})}{\alpha_{js}} \left[\sum_{i=1, i\neq k}^{N} \sum_{r=1}^{R} \alpha_{ir} p_{ir,js} \right] \end{aligned} \tag{15}$$

$$\begin{aligned} & \sum_{j\neq k} \sum_{s=1}^{R} \sum_{r=1}^{R} \sum_{y_0=1}^{N} \sum_{z_0=1}^{R} p(\mathbf{n} + e_{kr} - e_{js} - e_{kz_0}, \mathbf{y} + y_0, \mathbf{z} + z_0) \mu_{kr}(n_{kr}) p_{kr,js} \\ & = p(\mathbf{n}, \mathbf{y}, \mathbf{z}) \sum_{j\neq k} \sum_{s=1}^{R} \sum_{r=1}^{R} \sum_{z_0=1}^{R} \frac{\mu_{js}(n_{js})}{\alpha_{js}} \frac{\alpha_{kr}}{\alpha_{kz_0}} \frac{\mu_{kz_0}(n_{kz_0})}{\sum_{t=1}^{R} \mu_{kt}(n_{kt})} p_{kr,js} \left[\sum_{y_0=1}^{N} \sum_{t=1}^{R} \alpha_{y_0 t} p_{y_0 t, kz_0} \right] \\ & = p(\mathbf{n}, \mathbf{y}, \mathbf{z}) \sum_{j\neq k} \sum_{s=1}^{R} \frac{\mu_{js}(n_{js})}{\alpha_{js}} \sum_{r=1}^{R} \left[\alpha_{kr} p_{kr,js} \sum_{z_0=1}^{R} \left(\frac{\mu_{kz_0}(n_{kz_0})}{\sum_{t=1}^{R} \mu_{kt}(n_{kt})} \right) \right] \\ & = p(\mathbf{n}, \mathbf{y}, \mathbf{z}) \sum_{j\neq k} \sum_{s=1}^{R} \frac{\mu_{js}(n_{js})}{\alpha_{js}} \left[\sum_{r=1}^{R} \alpha_{kr} p_{kr,js} \right] \end{aligned} \tag{16}$$

By combining (15) and (16) we have

$$\begin{aligned} (15)+(16) &= p(\mathbf{n}, \mathbf{y}, \mathbf{z}) \sum_{j\neq k} \sum_{s=1}^{R} \frac{\mu_{js}(n_{js})}{\alpha_{js}} \left[\sum_{i=1, i\neq k}^{N} \sum_{r=1}^{R} \alpha_{ir} p_{ir,js} + \sum_{r=1}^{R} \alpha_{kr} p_{kr,js} \right] \\ &= p(\mathbf{n}, \mathbf{y}, \mathbf{z}) \sum_{j\neq k} \sum_{s=1}^{R} \frac{\mu_{js}(n_{js})}{\alpha_{js}} \left[\sum_{i=1}^{N} \sum_{r=1}^{R} \alpha_{ir} p_{ir,js} \right] \\ &= p(\mathbf{n}, \mathbf{y}, \mathbf{z}) \sum_{j\neq k} \sum_{s=1}^{R} \mu_{js}(n_{js}). \end{aligned} \tag{17}$$

In (14) we substitute the third term of (8) and obtain

$$\begin{aligned}
&\sum_{s=1}^{R} p\left(\mathbf{n}+e_{y_b s}, \mathbf{y}-y_b, \mathbf{z}-z_b\right) \mu_{y_b s}\left(n_{y_b s}+1\right) p_{y_b s, k z_b} \\
&= p(\mathbf{n}, \mathbf{y}, \mathbf{z}) \frac{\sum_{t=1}^{R} \mu_{kt}\left(n_{kt}\right)}{\sum_{t=1}^{R} \alpha_{y_b t} p_{y_b t, k z_b}} \sum_{s=1}^{R} \alpha_{y_b s} p_{y_b s, k z_b} \\
&= p(\mathbf{n}, \mathbf{y}, \mathbf{z}) \sum_{t=1}^{R} \mu_{kt}\left(n_{kt}\right). \qquad (18)
\end{aligned}$$

Thus, with (17) and (18) we have $p(\mathbf{n}, \mathbf{y}) \sum_{j=1}^{N} \sum_{s=1}^{R} \mu_{js}\left(n_{js}\right)$ which is equal to the right-hand side of (8). **QED**

Now we derive the throughput of the system $\lambda = \sum_{i=1}^{N} \lambda_i$, where λ_i is the throughput of station i, and the mean number of jobs $\overline{k}_i$ in station i.

Theorem 2. The *throughput* of station i is computed from

$$\begin{aligned}
\lambda_i = \sum_{(\mathbf{n},\mathbf{y},\mathbf{z}) \in S} p(\mathbf{n}, \mathbf{y}, \mathbf{z}) \Bigg[&\sum_{r=1}^{R} \mu_{ir}\left(n_{ir}\right) \left(1 - \sum_{k=1}^{N} \sum_{s=1}^{R} p_{ir,ks} I_{\{n_k = B_k\}}\right) \\
&+ \sum_{k=1}^{N} I_{\{n_k = B_k\}} I_{\{y_1 = i\}} \left(\sum_{r=1}^{R} \mu_{kr}\left(n_{kr}\right)\right)\Bigg] \qquad (19)
\end{aligned}$$

The *mean number of jobs* at station i is

$$\overline{k}_i = \sum_{(\mathbf{n},\mathbf{y},\mathbf{z}) \in S} p(\mathbf{n}, \mathbf{y}, \mathbf{z}) \left(\sum_{r=1}^{R} n_{ir} + \sum_{l=1}^{b} I_{\{y_l = i\}}\right). \qquad (20)$$

Proof. For any given state $(\mathbf{n}, \mathbf{y}, \mathbf{z})$ we consider the throughput of station i. Suppose the number of class r active jobs is n_{ir}, thus, the service rate is $\mu_{ir}\left(n_{ir}\right)$. However, a job which completed its service in station i can leave station i only by choosing a nonfull station j, i.e., $n_j < B_j$ with probability $\sum_{s=1}^{R} p_{ir,js}$. When considering the blocked jobs, if the first blocked job in station i is blocked by station k, i.e., $y_1 = i$, then this job has the same rate with which will leave station i and join the station k as the service rate of station k, i.e., $\sum_{r=1}^{R} \mu_{kr}\left(n_{kr}\right)$. Thus, (19) follows.

Similarly, for given a state $(\mathbf{n}, \mathbf{y}, \mathbf{z})$ we have the number of jobs in station i which will be equal to the active jobs $\sum_{r=1}^{R} n_{ir}$ plus the blocked jobs in this station, i.e., $\sum_{l=1}^{b} I_{\{y_l = i\}}$, so that (20) follows. **QED**

4 Central Server Model with Different Types of Stations

In this section we analyze central server models (star networks) with BAS mechanism. We put the constraint that the central server must have station type either I or IV and all other stations may

be of Type II or III. We further assume that only stations of Type I or IV will cause blocking, and the neighbors of such stations must be of Type II or III. The total number of jobs must satisfy the constraint $\min_{i\in T_1}\{B_i\} < K < \min_{i\in T_2}\{B_i\}$ and $p_{ir,js} = 0$, if $i, j \in T_1$.

We note that $p_{ir,js} = 0$ if $i, j \in T_1$ and for all r, s, then (1) will become:

$$\alpha_{js} = \sum_{i\in T_2}\sum_{r=1}^{R} \alpha_{ir} p_{ir,js} \qquad for \quad j \in T_1 \tag{21}$$

We denote T_A as the set of stations in T_1 if $n_i < B_i$, and T_B as the set of stations in T_1 with $n_i = B_i$. Note that $T_A \cup T_B = T_1$. In the following we give the global balance equation for a given state $(\mathbf{n}, \mathbf{x}, \mathbf{y}, \mathbf{z})$.

$$p(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z})\left(\sum_{i\in T_1}\mu_i(\mathbf{x}_i) + \sum_{i\in T_2}\sum_{r=1}^{R}\mu_{ir}(n_{ir})\right) \tag{22}$$

$$= \sum_{j\in T_A}\sum_{i\in T_2}\sum_{r=1}^{R} p\left(\mathbf{n}+e_{ir}, \mathbf{x}-x_{jn_j}, \mathbf{y}, \mathbf{z}\right)\mu_{ir}(n_{ir}+1)\, p_{ir,jx_{jn_j}}$$

$$+ \sum_{j\in T_B}\sum_{r=1}^{R} p\left(\mathbf{n}+e_{y_{jb_j}r}, \mathbf{x}, \mathbf{y}-y_{jb_j}, \mathbf{z}-z_{jb_j}\right)\mu_{y_{jb_j}r}\left(n_{y_{jb_j}r}+1\right) p_{y_{jb_j}r,jz_{jb_j}}$$

$$+ \sum_{i\in T_2, j\in T_2}\sum_{s=1}^{R}\sum_{r=1}^{R} p(\mathbf{n}+e_{ir}-e_{js}, \mathbf{x}, \mathbf{y}, \mathbf{z})\,\mu_{ir}(n_{ir}+1)\, p_{ir,js}$$

$$+ \sum_{i\in T_A, j\in T_2}\sum_{s=1}^{R}\sum_{x_{i0}=1}^{R} p(\mathbf{n}-e_{js}, \mathbf{x}+x_{i0}, \mathbf{y}, \mathbf{z})\,\mu_i(\mathbf{x}_i+x_{i0})\, p_{ix_{i0},js}$$

$$+ \sum_{i\in T_B, j\in T_2}\sum_{s=1}^{R}\sum_{y_{i0}=1}^{N}\sum_{x_{i0}=1}^{R} p(\mathbf{n}-e_{js}, \mathbf{x}+x_{i0}-x_{iB_i}, \mathbf{y}+y_{i0}, \mathbf{z}+z_{i0})\,\mu_i(\mathbf{x}_i+x_{i0}-x_{iB_i})\, p_{ix_{i0},js}.$$

We consider a given state $(\mathbf{n}, \mathbf{x}, \mathbf{y}, \mathbf{z})$ then the total flow-out rate from this state is $p(\mathbf{n}, \mathbf{x}, \mathbf{y}, \mathbf{z})\left(\sum_{i\in T_1}\mu_i(\mathbf{x}_i) + \sum_{i\in T_2}\sum_{r=1}^{R}\mu_{ir}(n_{ir})\right)$. In order to check the flow-in rates we consider the different events caused by these jobs: The first term on the right hand side of (22) denotes the departure event of a class r job from station $i \in T_2$ to station $j \in T_A$ to class x_{jn_j}. The second term denotes the events of a class r job in station $y_{jb_j} \in T_2$ which completed its service and is joining station $j \in T_B$. However, since station j is full, $n_j = B_j$, this job is blocked and becomes a class z_{jb_j} job and joins the list of blocked jobs waiting for station j. The third term denotes the event of moving of a class r job from station $i \in T_2$ to station $j \in T_2$ and becoming a class s job. The last two terms describing the events of moving of a class x_{i0} job from station $i \in T_A$, $i \in T_B$ to station $j \in T_2$ and becoming a class s job. We note that x_{i0} denotes the class in which the job leaves station i. y_{i0} and z_{i0} denote the station and the class of the blocked job at the first position of the list of blocked jobs which will move to station i. Note that $z_{i0} = x_{iB_i}$.

Theorem 3. The model has the following product form solution for equilibrium state probabilities:

$$p(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z}) = C \prod_{i\in T_1} f_i(\mathbf{x}_i,\mathbf{y}_i,\mathbf{z}_i) \prod_{i\in T_2} f_i(\mathbf{n}_i), \tag{23}$$

where $f_i(\mathbf{x}_i,\mathbf{y}_i,\mathbf{z}_i)$ for $i \in T_1$ is defined as

$$f_i(\mathbf{x}_i,\mathbf{y}_i,\mathbf{z}_i) = \begin{cases} \left(\frac{1}{\mu_i}\right)^{n_i} \prod_{l=1}^{n_i} \alpha_{ix_{il}}, & \text{if } i \text{ is Type } I \text{ and } i \in T_A \\ \left(\frac{1}{\mu_i}\right)^{n_i} \prod_{l=1}^{B_i} \alpha_{ix_{il}} \prod_{l=1}^{b_i} \frac{\sum_{t=1}^{R} \alpha_{y_{il}t} p_{y_{il}t,iz_{il}}}{\mu_i}, & \text{if } i \text{ is Type } I \text{ and } i \in T_B \\ \prod_{l=1}^{n_i} \frac{\alpha_{ix_{il}}}{\mu_{ix_{il}}}, & \text{if } i \text{ is Type } IV \text{ and } i \in T_A \\ \prod_{l=1}^{B_i} \frac{\alpha_{ix_{il}}}{\mu_{ix_{il}}} \prod_{l=1}^{b_i} \frac{\sum_{t=1}^{R} \alpha_{y_{il}t} p_{y_{il}t,iz_{il}}}{\mu_{ix_{il}z_{il}}}, & \text{if } i \text{ is Type } IV \text{ and } i \in T_B \end{cases} \tag{24}$$

and $f_i(\mathbf{n}_i)$ for $i \in T_2$, is defined as

$$f_i(\mathbf{n}_i) = \begin{cases} n_i! \prod_{r=1}^{R} \frac{1}{n_{ir}!} \left(\frac{\alpha_{ir}}{\mu_{ir}}\right)^{n_{ir}} & \text{if station } i \text{ is Type } II \\ \prod_{r=1}^{R} \frac{1}{n_{ir}!} \left(\frac{\alpha_{ir}}{\mu_{ir}}\right)^{n_{ir}} & \text{if station } i \text{ is Type } III \end{cases} \tag{25}$$

where C is a normalization constant

$$C^{-1} = \sum_{(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z})\in S} \prod_{i\in T_1} f_i(\mathbf{x}_i,\mathbf{y}_i,\mathbf{z}_i) \prod_{i\in T_2} f_i(\mathbf{n}_i). \tag{26}$$

Proof. From (23) – (25) we have

$$p\left(\mathbf{n}+e_{ir},\mathbf{x}-x_{jn_j},\mathbf{y},\mathbf{z}\right) = p(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z}) \frac{\mu_j(\mathbf{x}_j)}{\alpha_{jx_{jn_j}}} \frac{\alpha_{ir}}{\mu_{ir}(n_{ir}+1)} \tag{27}$$

$$\begin{aligned} &p\left(\mathbf{n}+e_{y_{jb_j}r},\mathbf{x}-x_{jn_j},\mathbf{y}-y_{jb_j},\mathbf{z}-z_{jb_j}\right) \\ &= p(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z}) \frac{\mu_j(\mathbf{x}_j)}{\sum_{t=1}^{R} \alpha_{y_{jb_j}t} p_{y_{jb_j}t,jz_{jb_j}}} \frac{\alpha_{y_{jb_j}r}}{\mu_{y_{jb_j}r}\left(n_{y_{jb_j}r}+1\right)} \end{aligned} \tag{28}$$

$$p(\mathbf{n}+e_{ir}-e_{js},\mathbf{x},\mathbf{y},\mathbf{z}) = p(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z}) \frac{\alpha_{ir}}{\mu_{ir(n_{ir}+1)}} \frac{\mu_{js(n_{js})}}{\alpha_{js}} \tag{29}$$

$$p(\mathbf{n}-e_{js},\mathbf{x}+x_{i0},\mathbf{y},\mathbf{z}) = p(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z}) \frac{\alpha_{ix_{i0}}}{\mu_i(\mathbf{x}_i+x_{i0})} \frac{\mu_{js(n_{js})}}{\alpha_{js}} \tag{30}$$

$$\begin{aligned} &p(\mathbf{n}-e_{js},\mathbf{x}+x_{i0}-x_{iB_i},\mathbf{y}+y_{i0},\mathbf{z}+z_{i0}) \\ &= p(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z}) \frac{\mu_{js(n_{js})}}{\alpha_{js}} \frac{\alpha_{ix_{i0}}}{\mu_i(\mathbf{x}_i+x_{i0}-x_{iB_i})} \frac{\sum_{t=1}^{R} \alpha_{y_{i0}t} p_{y_{i0}t,ix_{B_i}}}{\alpha_{ix_{iB_i}}} \frac{\mu_i(\mathbf{x}_i)}{\mu_i(\mathbf{z}_i+z_{i0})} \end{aligned} \tag{31}$$

Note that the last term on the right hand side of (31) is $\frac{\mu_i(\mathbf{x}_i)}{\mu_i(\mathbf{z}_i+z_{i0})} = 1$, because if the station i is of Type I, then the service rate is independent of the class. However, if the station is of Type

IV then $\mu_i(\mathbf{x}_i) = \mu_{iz_{iB_i}}$. Since we assume the blocked jobs are *FCFI*, then $\mu_i(\mathbf{z}_i + z_{i0}) = \mu_{iz_{i0}}$ and $x_{iB_i} = z_{i0}$. Now we substitute (27) – (31) on the right-hand side of (22) separately, and use (1) to obtain

$$\sum_{j\in T_A}\sum_{i\in T_2}\sum_{r=1}^{R} p\left(\mathbf{n}+e_{ir},\mathbf{x}-x_{jn_j},\mathbf{y},\mathbf{z}\right)\mu_{ir}(n_{ir}+1)\,p_{ir,jx_{jn_j}}$$

$$= p(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z})\sum_{j\in T_A}\sum_{i\in T_2}\sum_{r=1}^{R}\frac{\mu_j(\mathbf{x}_j)}{\alpha_{jx_{jn_j}}}\alpha_{ir}p_{ir,jx_{jn_j}} \tag{32}$$

$$= p(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z})\sum_{j\in T_A}\mu_j(\mathbf{x}_j) \tag{33}$$

$$\sum_{j\in T_B}\sum_{r=1}^{R} p\left(\mathbf{n}+e_{y_{jb_j}r},\mathbf{x},\mathbf{y}-y_{jb_j},\mathbf{z}-z_{jb_j}\right)\mu_{y_{jb_j}r}\left(n_{y_{jb_j}r}+1\right)p_{y_{j,b_j}r,jz_{jb_j}}$$

$$= p(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z})\sum_{j\in T_B}\sum_{r=1}^{R}\frac{\mu_j(\mathbf{x}_j)}{\sum_{t=1}^{R}\alpha_{y_{jb_j}t}p_{y_{jb_j}t,jz_{jb_j}}}\alpha_{y_{jb_j}r}p_{y_{j,b_j}r,jz_{jb_j}}$$

$$= p(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z})\sum_{j\in T_B}\mu_j(\mathbf{x}_j) \tag{34}$$

(35)

$$\sum_{i\in T_2}\sum_{j\in T_2}\sum_{s=1}^{R}\sum_{r=1}^{R} p(\mathbf{n}+e_{ir}-e_{js},\mathbf{x},\mathbf{y},\mathbf{z})\,\mu_{ir}(n_{ir}+1)\,p_{ir,js}$$

$$= p(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z})\sum_{j\in T_2}\sum_{s=1}^{R}\frac{\mu_{js}(n_{js})}{\alpha_{js}}\sum_{i\in T_2}\sum_{r=1}^{R}\alpha_{ir}p_{ir,js} \tag{36}$$

(37)

$$\sum_{i\in T_A}\sum_{j\in T_2}\sum_{s=1}^{R}\sum_{x_{i0}=1}^{R} p(\mathbf{n}-e_{js},\mathbf{x}+x_{i0},\mathbf{y},\mathbf{z})\,\mu_i(\mathbf{x}_i+x_{i0})\,p_{ix_{i0},js}$$

$$= p(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z})\sum_{j\in T_2}\sum_{s=1}^{R}\frac{\mu_{js}(n_{js})}{\alpha_{js}}\sum_{i\in T_A}\sum_{x_{i0}=1}^{R}\alpha_{ix_{i0}}p_{ix_{i0},js} \tag{38}$$

(39)

$$\sum_{i\in T_B}\sum_{j\in T_2}\sum_{s=1}^{R}\sum_{y_{i0}=1}^{N}\sum_{x_{i0}=1}^{R} p(\mathbf{n}-e_{js},\mathbf{x}+x_{i0}-x_{iB_i},\mathbf{y}+y_{i0},\mathbf{z}+z_{i0})\,\mu_i(\mathbf{x}_i+x_{i0}-x_{iB_i})\,p_{ix_{i0},js}$$

$$= p(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z}) \sum_{i\in T_B} \sum_{j\in T_2} \sum_{s=1}^{R} \sum_{x_{i0}=1}^{R} \frac{\mu_{js(n_{js})}}{\alpha_{js}} \frac{\alpha_{ix_{i0}} p_{ix_{i0},js}}{\alpha_{ix_{iB_i}}} \sum_{y_{i0}=1}^{N} \sum_{t=1}^{R} \alpha_{y_{i0}t} p_{y_{i0}t,ix_{iB_i}}$$

$$= p(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z}) \sum_{j\in T_2} \sum_{s=1}^{R} \frac{\mu_{js(n_{js})}}{\alpha_{js}} \left(\sum_{i\in T_B} \sum_{x_{i0}=1}^{R} \alpha_{ix_{i0}} p_{ix_{i0},js} \right) \tag{40}$$

By combining (36) – (40) we have

$$(36)+(38)+(40) = p(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z}) \sum_{j\in T_2} \sum_{s=1}^{R} \frac{\mu_{js(n_{js})}}{\alpha_{js}} \left(\sum_{i=1}^{N} \sum_{r=1}^{R} \alpha_{ir} p_{ir,js} \right)$$

$$\mu_{js(n_{js})}. \tag{41}$$

The result follows by checking the flow-in rate equal to the flow-out rate. **QED**

Theorem 4.

The *throughput* of the ith station is computed from

$$\lambda_i = \sum_{(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z})\in S} p(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z}) \begin{cases} \mu_i(\mathbf{x}_i) & \text{for } i \in T_1 \\ \sum_{r=1}^{R} \mu_{ir}(r_{ir}) \left(1 - \sum_{j\in T_B} \sum_{s=1}^{R} p_{ir,js}\right) & \text{for } i \in T_2 \\ \quad + \sum_{j\in T_B} \mu_j(\mathbf{x}_j) I_{\{y_{j1}=i\}}. \end{cases} \tag{42}$$

The *mean number of jobs* in the ith station is

$$\overline{k}_i = \sum_{(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z})\in S} p(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z}) \begin{cases} n_i & \text{for } i \in T_1 \\ \sum_{r=1}^{R} n_{ir} + \sum_{j\in T_B} \sum_{l=1}^{b_j} I_{\{y_{jl}=i\}} & \text{for } i \in T_2\,. \end{cases} \tag{43}$$

Proof. Consider a state $(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z})$. Since the jobs from stations of T_1 can never be blocked, thus the throughput of such station $i \in T_1$ is $\mu_i(\mathbf{x}_i)$. For stations in T_2 we have the throughput rate as the service rate $\sum_{r=1}^{R} \mu_{ir}(n_{ir})$. However, the job may choose the next station in T_B and a blocking may occur. If station $i \in T_2$ has blocked jobs which has the first priority to get into the destination station that caused their blocking, then the throughput rate of those jobs from station i is equal to the service rate of the station which caused blocking. Thus (42) follows.

The *mean number of jobs* is also obtained from given a state $(\mathbf{n},\mathbf{x},\mathbf{y},\mathbf{z})$. The number of jobs in station i for $i \in T_1$ is n_i. If $i \in T_2$, then the number of jobs in station i will include the active jobs of all classes $n_{i1} + \cdots + n_{iR}$ and the blocked jobs in that station. Thus, we take sum over the stations that cause blocking, $j \in T_B$, and check the vectors $\mathbf{y}_j$ if $y_{jl} = i$, then the l*th* job is located in station i, so that the result (42) follows. **QED**

5 Conclusions

We derived exact product form solution for the equilibrium state probabilities and computed throughput and mean number of jobs in two different models. We used a property of Type *II* and *III* stations that the locations of the blocked jobs has no effect on the service rates in those stations. Thus, we used this type of stations as kind of *optional* storage spaces for the blocking station. In other words, the system is treated as a virtual *nonblocking* system. By the well-defined state space, particularly the vector of the locations for the blocked jobs, we obtain the exact product form solutions. Another observation is that by the construction of the models we always have the constraint that no more than two jobs will move on a single departure event. The first model has a stronger constraint that is only one of the stations will cause blocking at a time. However, for the second model we allow more than one station to be of Type *I* or *IV* which may cause blocking at the same time. This is due to the constraint of the neighbors which will not cause blocking.

REFERENCES

[1] I. F. Akyildiz, "Exact Product Form Solution for Queueing Networks with Blocking", *IEEE Transactions on Computers*, Vol. C-36, No. 1, January 1987, pp. 122-125.

[2] I. F. Akyildiz, "On the Exact and Approximate Throughput Analysis of Closed Queueing Networks with Blocking", *IEEE Transactions on Software Engineering*, Vol. SE-14, No. 1, January 1988, pp. 62-71.

[3] I. F. Akyildiz and H. G. Perros, "Special Issue on Queueing Networks with Finite Capacity Queues: Introduction", *Performance Evaluation*, Vol. 10, No. 3, Dec. 1989.

[4] I. F. Akyildiz and H. von Brand, "Exact Solutions for Networks of Queues with Blocking-After-Service", *Technical Report, Georgia Tech, College of Computing, April 1991, GIT-COC-91-26.*

[5] I. F. Akyildiz and J. Liebeherr, "Optimal Deadlock Free Buffer Allocation in Multiple Chain Blocking Networks of Queues", *Proc. of the Int. Conference on the Performance of Distributed Systems and Integrated Communication Networks*, September 10-12, 1991.

[6] S. Balsamo, V. De Nitto Persone and G. Iazeoalla, "Identity and Reducibility Properties of Some Blocking and Non-blocking Mechanisms in Congested Networks", *Flow Control of Congested Networks*, NATO ASI Series, Springer Verlag, 1987, pp. 243-254.

[7] S. Balsamo and L. Donatiello, "On the Cycle Time Distribution in a Two-Stage Cyclic Network with Blocking", *IEEE Transactions on Software Engineering*, Vol. 15, No. 10, October 1989.

[8] F. Baskett, K. M. Chandy, R. R. Muntz and F. G. Palacios, "Open, Closed and Mixed Networks of Queues with Different Classes of Customers", *Journal of the ACM*, Vol. 22, Vol. 15, pp. 248-260, April 1975.

[9] P. P. Bocharov, "On the Two-Node Queucing Networks with Finite Capacity", *Proc. of the First Int. Workshop on Queueing Networks with Blocking*, North Holland, Dec. 1989, pp. 105-125.

[10] F. P. Kelly, "Networks of Queues with Customers of Different Types", *Journal of Applied Probability*, Vol. 12, pp. 542-554, 1975.

[11] F. P. Kelly, "Networks of Queues", *Advances in Applied Probability*, Vol. 8, No. 2, pp. 416-432, June 1976.

[12] R. O. Onvural, "A Note on the Product Form Solutions of Multiclass Closed Queueing Networks with Blocking", *Performance Evaluation*, Vol. 10, No. 3, December 1989, pp. 247-255.

[13] R. O. Onvural, "A Survey of Closed Queueing Networks with Finite Buffers", *ACM Computing Surveys*, Vol. 22, No. 2, June 1990, pp. 83-121.

[14] R. O. Onvural and H. G. Perros, "Some Equivalencies between Closed Queueing Networks with Blocking", *Performance Evaluation Journal*, Vol. 9, Dec. 1988, pp. 111-118.

Performance of Distributed Systems
and Integrated Communication Networks
T. Hasegawa, H. Takagi and Y. Takahashi (Editors)
Elsevier Science Publishers B.V. (North-Holland)

Optimal Deadlock Free Buffer Allocation in Multiple Chain Blocking Networks of Queues *

Ian F. Akyildiz and Jörg Liebeherr
College of Computing,
Georgia Institute of Technology,
Atlanta, Georgia 30332, U.S.A.

Abstract

Blocking can lead to a deadlock state due to the finite buffer capacities in queueing network models. In this paper deadlock freedom is investigated in multiple chain queueing networks with blocking, and necessary and sufficient conditions are given. An algorithm is presented for finding a deadlock free buffer allocation with the least number of buffers, thus the optimal allocation. The algorithm maps the queueing network into a directed graph. A procedure for finding a subset of all cycles in the directed graph is used to generate the conditions for deadlock freedom of the network. Finally, the optimal buffer allocation is obtained from these conditions by applying linear programming techniques.

1 Introduction

Queueing networks have increased their importance in performance evaluation of computer systems and communication networks in the last two decades. Numerous methods have been developed for the analysis of systems with infinite station capacities. Since in actual systems the resources have a finite capacity, queueing networks with blocking must be used for performance analysis. Each station of a queueing network with blocking can be thought of as a device with a finite length. The network is simply a set of arbitrarily linked stations. Blocking arises due to the limitations imposed by the capacity of these stations. In particular, blocking occurs when the flow of jobs through one station may be stopped for a moment if a destination station has reached its full capacity. The set of rules that dictate when a station becomes blocked or unblocked is commonly referred as the blocking mechanism. There are few blocking mechanisms that have been studied in the literature [1].

In this work we consider the so-called blocking-after-service (BAS), also referred as

*This work was supported by the National Science Foundation under Grant No. CCR-9011981.

type 1 or manufacturing blocking mechanism. A job upon service completion at a station i attempts to enter the destination station j. If station j is full at that moment, the job is forced to wait in station i's server until it can enter destination station j. The server remains blocked for this period of time. It cannot serve other jobs waiting in the queue.

Finite station capacities and blocking can introduce a deadlock situation. In a simple example, a deadlock may occur if a job which has finished its service at station i's server waits to join station j, whose capacity is full. That job is blocked in station i. Another job which has finished service at station j now wants to proceed to station i, whose capacity is also full. It blocks the jth station. Both jobs are waiting for each other. As a result a deadlock situation arises. There are two possible solutions for the deadlock problem: Either include a strategy to handle deadlocks in the model or simply restrict yourself to cases where deadlock is impossible. We select the second solution and find conditions under which a multiple chain blocking network is deadlock free. Kundu and Akyildiz [2] showed that queueing networks with a single routing chain are deadlock free if the number of jobs in the network is less than the capacity of the directed cycle with minimal capacity. However, the conditions for deadlock freedom with multiple routing chains become very complex due to the interactivities between the routing chains.

We demonstrate these interactivities in the network model given in Figure 1. There are 2 stations and 2 routing chains in the model. Let B_{ir} denote the buffer of chain r at station i (for $i, r = 1, 2$) and let K_r denote the total number of jobs in chain r ($r = 1, 2$). Let the parameters be given by:

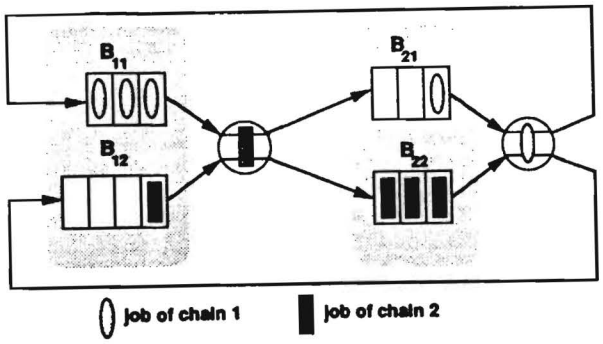


Figure 1: Blocking Network with 2 chains.

$$K_1 = 5; \quad K_2 = 6$$
$$B_{11} = 3; \quad B_{21} = 3$$
$$B_{12} = 4; \quad B_{22} = 3$$

Here a deadlock situation occurs, even though the condition for deadlock freedom given in [2] are satisfied for each routing chain in isolation.

In addition to determining the conditions for deadlock freedom another important issue is the allocation of station capacities in a queueing network such that deadlocks cannot occur.

Yilmazyildiz and Onozato [3] attempted to find conditions for deadlock freedom in multiple chain queueing networks with blocking. However, they failed to consider the interactions between different routing chains because they tried to extend the results of [2] straightforwardly. Therefore, they were not able to formalize the necessary conditions for deadlock freedom. Consequently, the algorithm in [3] for a deadlock free allocation of buffer capacities cannot guarantee optimality.

In this study, we provide the necessary and sufficient conditions for a deadlock free blocking network with multiple routing chains. In section 2 we describe the class of models which is considered in this study. The conditions for deadlock freedom are stated and proved in section 3. In section 4 we give a definition of an optimal buffer allocation. Then we present an algorithm to find an optimal allocation buffer capacities. Since the algorithm is not restricted to certain topologies, the algorithm generalizes the algorithm in [2] for the case of single chain networks. A crucial element of the allocation algorithm is an efficient method to identify cycles in a directed graph. This method is discussed in section 5. In section 6, we show applications of the allocation algorithm. We conclude our results in section 7.

2 Model Description

We consider a closed queueing network Γ with the following properties:

a) The system consists of N stations and R disjoint routing chains. Stations and routing chains are referred to by their indexes. Let $\mathcal{N}$ and $\mathcal{R}$ denote the set of stations and routing chains, respectively. Let $\mathcal{R}_i \subseteq \mathcal{R}$ denote the set of routing chains visiting station i. Let $\mathcal{N}_r \subseteq \mathcal{N}$ denote the set of stations visited by routing chain r.

b) Each station i has a single server. The service time distribution and the scheduling discipline of a station is arbitrary, but non-preemptive.

c) Each station keeps separate buffers for jobs from different routing chains. B_{ir} denotes the buffer at station i ($i \in \mathcal{N}$) excluding the server for jobs from routing chain r ($r \in \mathcal{R}_i$). Each buffer may accommodate only a finite number of jobs. Let Φ be an assignment of capacities to the buffers of Γ, i.e.,

$$\Phi : \{B_{ir} | i \in \mathcal{N}, \ r \in \mathcal{R}_i\} \rightarrow \{0, 1, 2, \ldots\} \tag{1}$$

Note that a buffer can have infinite capacity ($\Phi(B_{ir}) = \infty$) or no capacity at all ($\Phi(B_{ir}) = 0$).

d) The number of jobs in the system is fixed at

$$\underline{K} = (K_1, K_2, \ldots, K_R) \tag{2}$$

where

$$K_r = \sum_{i \in \mathcal{N}_r} k_{ir}, \qquad \text{for } r = 1, \ldots, R \tag{3}$$

represents the total number of jobs in routing chain r, and k_{ir} is the number of jobs of chain r in the i-th station (in buffer and servers). The state of the network is represented by a vector $\underline{k} = (\underline{k}_1, \underline{k}_2, \ldots, \underline{k}_N)$ where $\underline{k}_i$ is a vector $(k_{ir_1}, k_{ir_2}, \ldots, k_{ir_{|\mathcal{R}_i|}})$ with $r_u \in \mathcal{R}_i$ $(u = 1, 2, \ldots, |\mathcal{R}_i|)$. The total capacity of station i is computed by $\sum_{r \in \mathcal{R}_i} \Phi(B_{ir}) + 1$.

e) A job of routing chain r which has received service by station i proceeds to the station j with probability $p_{ij,r}$ (for $i, j \in \mathcal{N}$, $r \in \mathcal{R}_i$ and $r \in \mathcal{R}_j$). The number of jobs in the buffer of station j for jobs of chain r cannot exceed its capacity $\Phi(B_{jr})$. If B_{jr} is *saturated*, i.e., $k_{jr} = \Phi(B_{jr})$, the job of chain r is blocked at buffer B_{jr} and has to remain in the server of station i until a place in B_{jr} becomes available. This is the BAS (blocking-after-service) mechanism.

3 Conditions For Deadlock Freedom

In this section we present the deadlock freedom conditions for a queueing network as described in section 2. Before we state the theorem, we need the following definitions:

Definition 1 *A buffer cycle (of length $(t-1)$) $(1 \le t-1 \le |\mathcal{N}|)$ is a sequence of buffers $C = (B_{\sigma_1\rho_1}, B_{\sigma_2\rho_2}, \ldots, B_{\sigma_{t-1}\rho_{t-1}}, B_{\sigma_t\rho_t})$ such that*

$$\begin{aligned} &\sigma_u \in \mathcal{N} && (1 \le u \le t) \\ &\sigma_t = \sigma_1 \\ &\rho_u \in R_{\sigma_u}, \ \ \rho_{u+1} \in R_{\sigma_u}, \ \ p_{\sigma_u\sigma_{u+1},\rho_{u+1}} > 0 && (1 \le u < t). \end{aligned} \tag{4}$$

Let $\mathcal{C}$ denote the set of all buffer cycles in Γ.

Definition 2 *The set of stations with buffers from routing chain r in buffer cycle C is defined by*

$$S_r^{(C)} = \{i \mid B_{ir} \in C, i \in \mathcal{N}\} \tag{5}$$

Theorem 1 *A multiple chain queueing network Γ is* deadlock free *(DLF) if and only if for all $C \in \mathcal{C}$ with $C = (B_{\sigma_1\rho_1}, B_{\sigma_2\rho_2}, \ldots, B_{\sigma_{t-1}\rho_{t-1}}, B_{\sigma_t\rho_t})$ there exists a routing chain $r \in \mathcal{R}$ such that:*

$$\sum_{1 < u \le t \wedge \sigma_u \in S_r^{(C)}} \Phi(B_{\sigma_u r}) + |S_r^{(C)}| > K_r \tag{6}$$

Therefore, in each buffer cycle there must be at least one routing chain which cannot both saturate the buffers of its chain in the cycle and have blocked jobs waiting at the saturated buffers.

Proof:

1. *Necessity:*
   Assume there exists a cycle $\hat{C} = (B_{\sigma_1\rho_1}, B_{\sigma_2\rho_2}, \ldots, B_{\sigma_{\hat{t}-1}\rho_{\hat{t}-1}}, B_{\sigma_{\hat{t}}\rho_{\hat{t}}})$ with:

   $$(\forall r \in \mathcal{R}) \ \Big(\sum_{1 < u \le \hat{t} \wedge \sigma_u \in S_r^{(\hat{C})}} \Phi(B_{\sigma_u r}) + |S_r^{(C)}| \le K_r \Big). \tag{7}$$

   Then, Γ has a feasible state where all buffers in $\hat{C}$ are saturated, i.e., $B_{\sigma_u\rho_u}$ contains $\Phi(B_{\sigma_u\rho_u})$ jobs of chain ρ_u $(1 \le u < \hat{t})$. Additionally, the server of station σ_u may contain a job of chain ρ_{u+1}. Note that the entire network will have $|S_r^{(C)}|$ servers which contain a job of chain r $(r \in \mathcal{R})$. There exists a positive probability that the job in station σ_u's server has picked station σ_{u+1} as its next destination. In this state, no server can release a job and eventually, each station σ_u is blocked. Since the scheduling is non-preemptive this state will persist and a deadlock state is reached. □

2. *Sufficiency:*
   Assume the multiple chain queueing network with allocation Φ is in a deadlock state. Then there must exist a permanently blocked station, say σ_1, which holds a job from routing chain, say ρ_2 $(\rho_2 \in \mathcal{R}_{\sigma_1})$, in its server. The job is blocked at a saturated buffer, say $B_{\sigma_2\rho_2}$ of some station, say σ_2, i.e., $k_{\sigma_2\rho_2} = \Phi(B_{\sigma_2\rho_2})$. Station σ_2 itself must be blocked, otherwise a space in $B_{\sigma_2\rho_2}$ will eventually become available and station σ_1 would not be blocked anymore. The job in the server of station σ_2 may be blocked at a saturated buffer, say $B_{\sigma_3\rho_3}$ of some station, say σ_3, i.e., $k_{\sigma_3\rho_3} = \Phi(B_{\sigma_3\rho_3})$. We can continue to apply this argument. Because of the finiteness of stations and buffers, we will encounter a station σ_{t-1} which is blocked at a buffer of station σ_1, say $B_{\sigma_1\rho_1}$. Then, buffers $B_{\sigma_2\rho_2}, B_{\sigma_3\rho_3}, \ldots, B_{\sigma_1\rho_1}$ define a buffer cycle $\hat{C} = (B_{\sigma_1\rho_1}, B_{\sigma_2\rho_2}, \ldots, B_{\sigma_{i-1}\rho_{i-1}}, B_{\sigma_i\rho_i})$ $(B_{\sigma_1\rho_1} = B_{\sigma_i\rho_i})$. From $\hat{C}$ we construct the sets $S_r^{(\hat{C})}$. Note that $|S_r^{(\hat{C})}|$ jobs of chain r are blocked in servers of stations which have a buffer in the cycle. If we sum up the number of jobs which are in buffers and servers of the cycle for each chain r, we obtain for each chain:

   $$\sum_{1 < u \le i \wedge \sigma_u \in S_r^{(\hat{C})}} \Phi(B_{\sigma_u r}) + |S_r^{(\hat{C})}| \tag{8}$$

   The sum of each chain r is less or equal the total number of jobs K_r. Therefore:

   $$\sum_{1 < u \le i \wedge \sigma_u \in S_r^{(\hat{C})}} \Phi(B_{\sigma_u r}) + |S_r^{(\hat{C})}| < K_r \tag{9}$$

   However, this is a violation of the DLF condition (equation (6)) for cycle $\hat{C}$. □

The following Corollary follows directly from Theorem 1:

Corollary 1 *The DLF condition is satisfied for a cycle $C \in \mathcal{C}$, if there exists an $r \in \mathcal{R}$ with:*

$$|S_r^{(C)}| > K_r \tag{10}$$

In other words, if the number of stations with buffers from a routing chain in a cycle exceeds the total number of jobs of that chain, then the deadlock freedom condition for that cycle is satisfied disregarding the capacities of the buffers.

If the DLF conditions (equation (6)) are satisfied for a buffer cycle C, they are clearly satisfied for all cycles which include the buffers of cycle C as a subset. Therefore, it is not required to test the DLF conditions for all buffer cycles in order to guarantee deadlock freedom of the network. This is formalized in the following Lemma:

Lemma 1 *Given a multiple chain queueing network Γ and the set of all buffer cycles $\mathcal{C}$ of Γ. For any $C', C'' \in \mathcal{C}$ let $C' \subseteq C''$, if the set of buffers in C' is a subset of the set of buffers in C''. Let $\tilde{\mathcal{C}} \subseteq \mathcal{C}$ be such that*

$$(\forall C', C'' \in \tilde{\mathcal{C}})(C' \not\subseteq C''). \tag{11}$$

Then, Γ is deadlock free if the DLF condition (equation (6)) is satisfied for all $C \in \tilde{\mathcal{C}}$.

Proof: If $C' \subseteq C''$, then $S_r^{(C')} \subseteq S_r^{(C'')}$ for all $r \in \mathcal{R}$. Therefore, if the DLF condition is satisfied for a routing chain r in C', i.e., $\sum_{1<u\le t \wedge \sigma_u \in S_r^{(C')}} \Phi(B_{\sigma_u r}) + |S_r^{(C)}| > K_r$, it clearly is satisfied for any superset of C'. □

4 Deadlock Free Buffer Allocation Algorithm

Once the conditions for a deadlock free network are known, we can investigate the problem of finding DLF allocation with the least number of buffers.

Definition 3 *A buffer allocation Φ^* for a queueing network Γ is optimal if:*

(i) Φ^* satisfies the DLF condition (equation (6)) for all $C \in \mathcal{C}$,

(ii) Φ^* is minimal, i.e., no deadlock free allocation Φ for Γ allocates less capacities to the buffers of Γ than Φ^*:

$$\sum_{i\in\mathcal{N}} \sum_{r\in\mathcal{R}_i} \Phi^*(B_{ir}) \le \sum_{i\in\mathcal{N}} \sum_{r\in\mathcal{R}_i} \Phi(B_{ir}) \tag{12}$$

In the remaining part of this section we will present an algorithm which finds an optimal buffer allocation for a queueing network Γ. The algorithm is executed in two steps:

1. Find the set $\tilde{\mathcal{C}}$ of buffer cycles in Γ and establish the sets $S_r^{(C)}$ ($r \in \mathcal{R}$) for eac $C \in \tilde{\mathcal{C}}$.

2. Obtain a linear program from $S_r^{(C)}$ ($r \in \mathcal{R}$, $C \in \tilde{\mathcal{C}}$). Solve the linear program wit respect to the objective that the total number of allocated buffers be minimal.

In the following subsections we will discuss these steps in detail.

4.1 Finding Buffer Cycles

We approach the problem of finding the buffer cycles of a queueing network as the problen of finding so-called elementary cycles in a directed graph. The following Lemma allow to map the queueing network into a directed graph:

Lemma 2 *Given a multiple chain queueing network Γ.*
Obtain a directed graph $G_\Gamma = (V_\Gamma, E_\Gamma)$ with a set of vertices V_Γ and a set of arcs E_Γ fron Γ by:

(i) $V_\Gamma = \{B_{ir} |\ i \in \mathcal{N},\ r \in \mathcal{R}_i\}$,

(ii) $E_\Gamma = \{(B_{ir}, B_{js}) |\ r \in \mathcal{R}_i \ \wedge\ (\exists s \in \mathcal{R})(s \in \mathcal{R}_i \ \wedge\ s \in \mathcal{R}_j \ \wedge\ p_{ij,s} > 0)\}$.

Then, C is a buffer cycle in Γ if and only if C is a cycle in G_Γ.

Proof: The proof follows immediately from the definition of a buffer cycle and the construction of the directed graph. □

To obtain the list of cycles in a directed graph, we will apply an algorithm which is based on a method to find elementary cycles in a directed graph [4]. The algorithm in [4], referred to as *Johnson's algorithm*, generates a complete list of elementary cycles. However, we are not interested in all cycles of G_Γ. Rather, motivated by Lemma 1, we want to find the set of cycles $\tilde{\mathcal{C}}$ where no two cycles are a subset of each other. This restriction allows to improve the cycle finding algorithm significantly, compared to the original version of *Johnson's algorithm*. We discuss the details of the algorithm in section 5.

Corollary 1 states that the DLF condition (equation (6)) is satisfied for certain buffer cycles disregarding the capacities which are allocated to the buffers of the cycle. Therefore, buffer cycles which satisfy the condition in Corollary 1 (equation (10)) can be eliminated in the forthcoming steps of the algorithm. Note however, that the elimination is dependent on the total number of jobs in each chain K_r.

4.2 Optimization with Linear Programming

The algorithm presented in the previous subsections provides us with the set of buffer cycles $\tilde{\mathcal{C}}$. These cycles need to be examined to determine deadlock freedom of an allocation

Φ. If the deadlock condition is satisfied for all buffer cycles in $\tilde{\mathcal{C}}$, then the network is deadlock free. Finding an optimal buffer allocation Φ^* for a given network Γ is the solution of the following optimization problem:

$$\begin{aligned}&\text{Find } \Phi^* \text{ which minimizes} \sum_{i\in\mathcal{N}}\sum_{r\in\mathcal{R}_i} \Phi^*(B_{ir}) \text{ with the constraints:}\\ &(\forall C \in \tilde{\mathcal{C}})(\exists r \in \mathcal{R})\ (C = (B_{\sigma_1\rho_1}, B_{\sigma_2\rho_1}, \ldots, B_{\sigma_{t-1}\rho_{t-1}}, B_{\sigma_t\rho_t})\\ &\quad\rightarrow \sum_{1<u\le t \wedge \sigma_u \in S_r^{(C)}} \Phi(B_{\sigma_u r}) + |S_r^{(C)}| > K_r)\end{aligned} \tag{13}$$

Let the elements of $\tilde{\mathcal{C}}$ be ordered, i.e., $\tilde{\mathcal{C}} = \{C_1, C_2, \ldots, C_{|\tilde{\mathcal{C}}|}\}$, and let t_l denote the number of buffers in cycle C_l. In the following, we denote $\Phi(B_{ir})$ by b_{ir} in order to stress that the allocation of capacity to a buffer is a variable for the optimization process. Then, the optimization problem can be written as:

$$\begin{aligned}&\textbf{minimize} \quad \sum_{i\in\mathcal{N}}\sum_{r\in\mathcal{R}_i} b_{ir}\\ &\textbf{subject to}\\ &\quad \bigvee_{S_r^{(C_l)}\neq\emptyset} \sum_{1<u\le t_l \wedge \sigma_u \in S_r^{(C_l)}} b_{\sigma_u r} + |S_r^{(C_l)}| > K_r \quad (l = 1,2,\ldots,|\tilde{\mathcal{C}}|)\\ &\quad b_{ir} \ge 0 \qquad\qquad (i \in \mathcal{N},\ r \in \mathcal{R}_i).\end{aligned} \tag{14}$$

By introducing additional $(0,1)$-variables which allow to replace the disjunctions in the constraints, we can state the optimization problem as a linear program:

$$\begin{aligned}&\textbf{minimize} \quad \sum_{i\in\mathcal{N}}\sum_{r\in\mathcal{R}_i} b_{ir}\\ &\textbf{subject to}\\ &\quad \sum_{1<u\le t_l \wedge \sigma_u \in S_r^{(C_l)}} b_{\sigma_u r} - y_{lr}\cdot(K_r - S_r^{(C_l)} + 1) \ge 0\\ &\qquad\qquad (l = 1,2,\ldots,|\tilde{\mathcal{C}}|;\ S_r^{(C_l)} \neq \emptyset)\\ &\quad \sum_{S_r^{(C_l)}\neq\emptyset} y_{lr} \ge 1 \qquad (l = 1,2,\ldots,|\tilde{\mathcal{C}}|)\\ &\quad b_{ir} \ge 0 \qquad\qquad (i \in \mathcal{N},\ r \in \mathcal{R}_i)\\ &\quad y_{lr} \in \{0,1\} \qquad (l = 1,2,\ldots,|\tilde{\mathcal{C}}|;\ S_r^{(C_l)} \neq \emptyset)\end{aligned} \tag{15}$$

Note that y_{lr} is not needed for cycles which contain buffers from only one routing chain, i.e.,

$$(\exists r \in \mathcal{R})(\forall r' \in \mathcal{R})(r \neq r' \rightarrow S_r^{(C_l)} = \emptyset). \tag{16}$$

The optimization of the equation system in (15) can be solved with any program package for integer linear programs. The solution of the linear program provides values for b_{ir} which are an optimal buffer allocation Φ^* for network Γ. Note that in general, the system will have more than one optimal solution, since a linear problem with the given structure will have considerable degeneracy.

5 Cycle Finding Algorithm

5.1 Algorithm

Here we discuss the algorithm which finds the set of cycles $\tilde{\mathcal{C}}$ *in a directed graph wi the constraint that no two cycles are a subset of each other. The following terminolo is needed for the discussion:

Definition 4 *Given a directed graph $G = (V, E)$.*

- *A cycle in G is elementary, if no vertex but the first and the last appear twice.*
- *G is strongly connected, if for any two vertices u and v there is a path from u to*
- *A strongly connected component (SCC) of a directed graph G is maximal strong connected subgraph.*
- *The adjacency list $A_G(u)$ ($u \in V$) of G contains v if $(u,v) \in E$. The adjacen structure A_G is the list of all adjacency lists $A_G(u)$ for $u \in V$.*

For our purposes we extended *Johnson's algorithm* (see section 4.2) to achieve a bett performance for the cycle finding procedure. A simple extension is a fast search fo Single-Vertex and Two-Vertex Cycles, i.e., cycles which involve only one vertex or tw vertices:

- Single-Vertex cycles are not considered in [4]. However, they do frequently occur i a graph obtained from a queueing network. Therefore, we added a feature whic looks for single vertex cycles and eliminates a vertex v from G, if a cycle (v,v) exists

$$\begin{aligned}&V := V \setminus \{v\}\\ &A_G := A_G \setminus \{A_G(v)\}\\ &A_G(w) := A_G(w) \setminus \{v\} \quad (\text{for } w \in V_G).\end{aligned} \tag{17}$$

- A Two-Vertex cycle, i.e., a cycle (v,w,v) $(v,w \in V)$, can be easily identified from the adjacency structure. Assuming that vertices with a Single-Vertex cycle have been removed from the graph, any cycle along an arc (v,w) or (w,v) is a superset of a Two-Vertex cycle (v,w,v). Since Lemma 1 allows to disregard the supersets of any cycle without violating the DLF conditions, we eliminate the edges (v,w) or (w,v) from the graph after a cycle (v,w,v) has been detected, i.e.:

$$A_G(w) := A_G(w) \setminus \{v\} \qquad A_G(v) := A_G(v) \setminus \{w\}. \tag{18}$$

*Since buffer cycles in Γ can be mapped into cycles of graph G_Γ (Lemma 4.1), we use $\mathcal{C}$ to denote the set of cycles in a directed graph.

If the set of vertices in a path of a cycle C is a superset of the set of vertices of a cycle C', then $C \subseteq C'$. Therefore, we are able to disregard cycles before we have complete knowledge of the set of vertices in a cycle. Since cycles are developed along a path, we included a procedure which compares the current path with the set of already detected cycles. As we will see at the end of this section, the improvement in running time is mainly due to this extension.
A complete description of the extended version of *Johnson's algorithm* is given in the Appendix.

The algorithm finds cycles by constructing paths where no vertex appears twice. The current path is stored on a stack. Calling CIRCUIT() appends a vertex to the stack, a return from CIRCUIT() causes the vertex on top of the stack to be removed. To prevent a vertex from being added more than once to the stack, a vertex is blocked when CIRCUIT() is called. Unblocking of vertices is done before a return from CIRCUIT(), by calling procedure UNBLOCK(). Procedure CIRCUIT() returns value TRUE if a cycle has been found on the current stack, and FALSE otherwise. The details of procedures CIRCUIT() and UNBLOCK() are discussed in [4].

Procedure SIMPLE_CYCLE() detects Single–Vertex and Two–Vertex cycles as discussed earlier in this section.

Each time a cycle is found, it is added to a list. Note that we are not interested in finding all cycles of the graph. Rather, we want to find only those cycles which are not a subset of each other (Lemma 1). Therefore, before a cycle is added to the list of relevant cycles, we call a procedure UPDATE_CYCLE() which checks if the new cycle is a superset set of an already included cycle. In this case the new cycle can be disregarded. If the new cycle is a subset of some existing cycles, these cycles are eliminated and the new cycle is included into the list.

Procedure CHECK_STACK() tests the current stack against the existing list of cycles. If the contents of the stack is a superset of any already detected cycle, the search along the current stack is interrupted.

5.2 Evaluation of the Algorithm

The following examples will demonstrate how the performance of *Johnson*'s algorithm is improved with the presented extensions. We compare the discussed algorithm with a version where the procedures SIMPLE_CYCLE() and CHECK_STACK() are not included. This version corresponds to the original *Johnson*'s algorithm. We show examples of directed graphs where the adjacency structure is generated randomly with $Prob[(u,v) \in E] = p$ $(u,v \in V, 0 < p < 1)$. We do not consider Single–Vertex cycles, since Single–Vertex cycles result in a deletion of a vertex. Table 1 and Table 2 show the results for a graph with 10 and 20 vertices, respectively. The first column of the table provides the value of probability p, the second column the number of cycles in set $\tilde{C}$. Note that both algorithms provide the same output, i.e., the set $\tilde{C}$. The last two columns show the results for *Johnson*'s algorithm and the new version of the cycle finding algorithm. For each version we provide the number of cycles in the graph which are generated during the execution of the algorithm and the running times of each algorithm. We terminated program runs, if their CPU time exceeded 10^5 seconds. The algorithm is implemented in C and run on a Sun Sparc workstation. We see in Tables 1 and 2 that the running time of *Johnson*'s algorithm increases fast and becomes soon computationally impractical. Note that Johnson's algorithm wastes the running time by examining a very high number of cycles even when the size of set $\tilde{C}$ is small.

$\lvert V\rvert = 10$			Johnson	New
$p = 0.2$	$\lvert\tilde{C}\rvert = 1$	Cycles examined:	1	1
		CPU time (*sec*):	< 0.1	< 0.1
$p = 0.4$	$\lvert\tilde{C}\rvert = 12$	Cycles examined:	520	16
		CPU time (*sec*):	0.2	< 0.1
$p = 0.6$	$\lvert\tilde{C}\rvert = 21$	Cycles examined:	15,287	22
		CPU time (*sec*):	9.0	< 0.1
$p = 0.8$	$\lvert\tilde{C}\rvert = 27$	Cycles examined:	117,004	27
		CPU time (*sec*):	36.5	< 0.1

Table 1: Finding Cycles in a Graph with 10 Vertices.

$\lvert V\rvert = 20$			Johnson	New
$p = 0.2$	$\lvert\tilde{C}\rvert = 17$	Cycles examined:	3,718	39
		CPU time (*sec*):	3.4	0.2
$p = 0.4$	$\lvert\tilde{C}\rvert = 64$	Cycles examined:	—	70
		CPU time (*sec*):	$> 10^5$	0.2
$p = 0.6$	$\lvert\tilde{C}\rvert = 113$	Cycles examined:	—	127
		CPU time (*sec*):	$> 10^5$	0.4
$p = 0.8$	$\lvert\tilde{C}\rvert = 129$	Cycles examined:	—	130
		CPU time (*sec*):	$> 10^5$	0.1

Table 2: Finding Cycles in a Graph with 20 Vertices.

6 Examples

In this section we show applications of the deadlock free buffer allocation algorithm. We give two examples, one for a single chain queueing network, and another for a multiple chain queueing network. All linear programs shown in the examples were solved with the programming package LINDO [5].

6.1 Example 1

Note that the deadlock free buffer allocation algorithm for single chain queueing networks in [2] is only applicable to so-called cacti networks, i.e., queueing networks where no two cycles have more than one buffer in common. Our algorithm does not have any restrictions on the structure of the network. We will show the allocation algorithm for the network shown in Figure 2. Since we have a single chain network, the chain identifying index

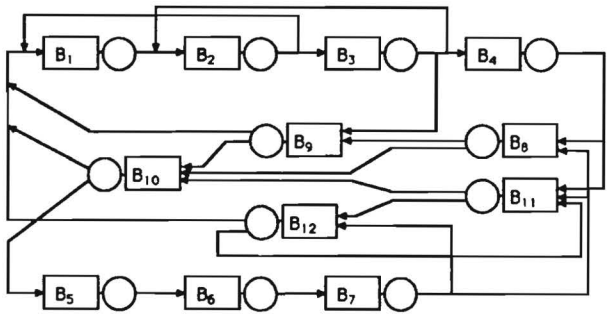


Figure 2: Single Chain Queueing Network.

has been omitted, i.e., B_i denotes the single buffer of station i. Although the networks contains 28 elementary cycles, the cycle finding algorithm in section 5 produces only 7 cycles which are relevant for a deadlock free allocation. The cycles are:

$$\begin{aligned} C_1 &= (B_1, B_2, B_1) \\ C_2 &= (B_2, B_3, B_2) \\ C_3 &= (B_3, B_4, B_3) \\ C_4 &= (B_5, B_6, B_7, B_8, B_{10}, B_5) \\ C_5 &= (B_5, B_6, B_7, B_{10}, B_{11}, B_5) \\ C_6 &= (B_5, B_6, B_7, B_{12}, B_5) \\ C_7 &= (B_{11}, B_{12}, B_{11}) \end{aligned}$$

The linear program corresponding to the set of cycles has the following structure. Recall from section 4.2 that we use b_i to denote the variable for the capacity assigned to buffer B_i.

$$\text{minimize } b_1 + b_2 + b_3 + b_4 + b_5 + b_6 + b_7 + b_8 + b_9 + b_{10} + b_{11} + b_{12}$$

subject to

$$\begin{aligned} b_1 + b_2 &\geq K - 1 \\ b_2 + b_3 &\geq K - 1 \\ b_3 + b_4 &\geq K - 1 \\ b_5 + b_6 + b_7 + b_8 + b_{10} &\geq K - 4 \\ b_5 + b_6 + b_7 + b_{10} + b_{11} &\geq K - 4 \\ b_5 + b_6 + b_7 + b_{12} &\geq K - 3 \\ b_{11} + b_{12} &\geq K - 1 \\ b_i &\geq 0 \qquad (i = 1, 2, \ldots, 12). \end{aligned}$$

Note that we do not need (0,1)-variables y_{l_r} for single chain networks. Table 3 presents the solution of the linear program for different values for the total number of jobs K. B^* denotes the sum of buffer allocations. We obtain an optimal deadlock free buffer allocation Φ^* for the network in Figure 2 with:

$$\Phi^*(B_i) = b_i \quad (i = 1, 2, \ldots, 12)$$

K	4	10	35
B^*	9	33	133
b_1	0	0	0
b_2	3	9	34
b_3	0	0	0
b_4	3	9	34
b_5	0	2	0
b_6	0	0	0
b_7	0	0	0
b_8	0	4	0
b_9	0	0	0
b_{10}	0	0	31
b_{11}	0	4	0
b_{12}	3	5	34

Table 3: Results for Example 1.

For $K = 4$, the condition of Corollary 1 (equation (10) is satisfied for the following cycles:

$$\begin{aligned} C_4 &= (B_5, B_6, B_7, B_8, B_{10}, B_5) \\ C_5 &= (B_5, B_6, B_7, B_{10}, B_{11}, B_5) \\ C_6 &= (B_5, B_6, B_7, B_{12}, B_5) \end{aligned}$$

Therefore, we can eliminate the constraints in the linear program which were derived from these cycles.

6.2 Example 2

Figure 3 depicts a queueing network with three routing chains. In the first step of the

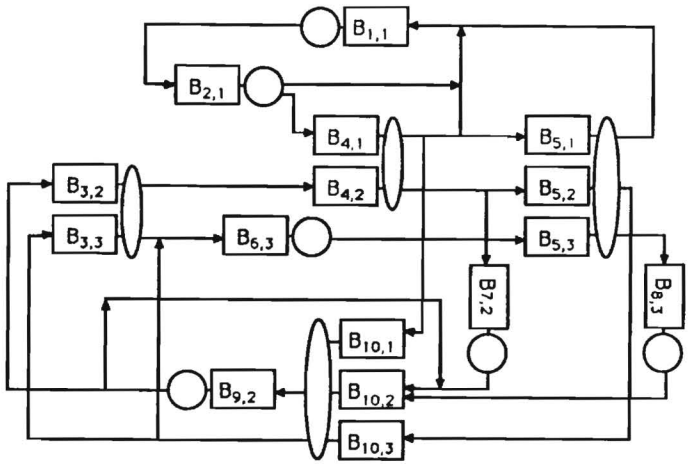


Figure 3: Multiple Chain Queueing Network.

algorithm, we construct a directed graph according to Lemma 2. The directed graph has 429 elementary cycles. However, the algorithm described in section 5 yields only 19 cycles which are relevant for the optimization:

$$
\begin{aligned}
C_1 &= (B_{1,1}, B_{2,1}, B_{1,1}) \\
C_2 &= (B_{9,2}, B_{10,2}, B_{9,2}) \\
C_3 &= (B_{5,3}, B_{8,3}, B_{10,3}, B_{6,3}, B_{5,3}) \\
C_4 &= (B_{2,1}, B_{4,1}, B_{10,1}, B_{2,1}) \\
C_5 &= (B_{2,1}, B_{4,1}, B_{5,1}, B_{10,2}, B_{2,1}) \\
C_6 &= (B_{2,1}, B_{4,1}, B_{5,2}, B_{8,3}, B_{10,3}, B_{2,1}) \\
C_7 &= (B_{2,1}, B_{4,1}, B_{5,2}, B_{10,2}, B_{2,1}) \\
C_8 &= (B_{2,1}, B_{4,1}, B_{7,2}, B_{10,2}, B_{2,1}) \\
C_9 &= (B_{4,2}, B_{5,1}, B_{8,3}, B_{10,3}, B_{3,3}, B_{4,2}) \\
C_{10} &= (B_{4,2}, B_{5,1}, B_{8,3}, B_{10,3}, B_{9,2}, B_{3,2}, B_{4,2}) \\
C_{11} &= (B_{4,2}, B_{5,1}, B_{10,2}, B_{3,3}, B_{4,2})
\end{aligned}
$$

$$
\begin{aligned}
C_{12} &= (B_{4,2}, B_{5,2}, B_{8,3}, B_{10,3}, B_{3,3}, B_{4,2}) \\
C_{13} &= (B_{4,2}, B_{5,2}, B_{8,3}, B_{10,3}, B_{9,2}, B_{3,2}, B_{4,2}) \\
C_{14} &= (B_{4,2}, B_{5,2}, B_{10,2}, B_{3,3}, B_{4,2}) \\
C_{15} &= (B_{4,2}, B_{10,1}, B_{3,3}, B_{4,2}) \\
C_{16} &= (B_{4,2}, B_{10,1}, B_{9,2}, B_{3,2}, B_{4,2}) \\
C_{17} &= (B_{4,2}, B_{7,2}, B_{10,2}, B_{3,3}, B_{4,2}) \\
C_{19} &= (B_{5,3}, B_{10,2}, B_{6,3}, B_{5,3})
\end{aligned}
$$

To write the linear program we first have to establish the sets $S_r^{(C_l)}(r = 1,2,3;\ l = 1,2,\ldots,19)$. Then we obtain:

$$
\begin{aligned}
\textbf{minimize} \quad & b_{1,1} + b_{2,1} + b_{4,1} + b_{5,1} + b_{10,1} \\
& + b_{3,2} + b_{4,2} + b_{5,2} + b_{7,2} + b_{9,2} + b_{10,2} \\
& + b_{3,3} + b_{5,3} + b_{6,3} + b_{8,3} + b_{10,3}
\end{aligned}
$$

subject to

$$
\begin{aligned}
C_1: \quad & b_{1,1} + b_{2,1} \geq K_1 - 1 \\
C_2: \quad & b_{9,2} + b_{10,2} \geq K_2 - 1 \\
C_3: \quad & b_{5,3} + b_{6,3} + b_{8,3} + b_{10,3} \geq K_3 - 3 \\
C_4: \quad & b_{2,1} + b_{4,1} + b_{10,1} \geq K_1 - 2 \\
C_5: \quad & b_{2,1} + b_{4,1} + b_{5,1} - (K_1 - 2) \cdot y_{5,1} \geq 0 \\
& b_{10,2} - K_2 \cdot y_{5,2} \geq 0 \\
& y_{5,1} + y_{5,2} \geq 1 \\
C_6: \quad & b_{2,1} + b_{4,1} - (K_1 - 1) \cdot y_{6,1} \geq 0 \\
& b_{5,2} - K_2 \cdot y_{6,2} \geq 0 \\
& b_{8,3} + b_{10,3} - (K_3 - 1) \cdot y_{6,3} \geq 0 \\
& y_{6,1} + y_{6,2} + y_{6,3} \geq 1 \\
C_7: \quad & b_{2,1} + b_{4,1} - (K_1 - 1) \cdot y_{7,1} \geq 0 \\
& b_{5,2} + b_{10,2} - (K_2 - 1) \cdot y_{7,2} \geq 0 \\
& y_{7,1} + y_{7,2} \geq 1 \\
C_8: \quad & b_{2,1} + b_{4,1} - (K_1 - 1) \cdot y_{8,1} \geq 0 \\
& b_{7,2} + b_{10,2} - (K_2 - 1) \cdot y_{8,2} \geq 0 \\
& y_{8,1} + y_{8,2} \geq 1 \\
C_9: \quad & b_{5,1} - K_1 \cdot y_{9,1} \geq 0 \\
& b_{4,2} - K_2 \cdot y_{9,2} \geq 0 \\
& b_{3,3} + b_{8,3} + b_{10,3} - (K_3 - 2) \cdot y_{9,3} \geq 0 \\
& y_{9,1} + y_{9,2} + y_{9,3} \geq 1 \\
C_{10}: \quad & b_{5,1} - K_1 \cdot y_{10,1} \geq 0 \\
& b_{4,2} + b_{3,2} + b_{9,2} - (K_2 - 2) \cdot y_{10,2} \geq 0 \\
& b_{8,3} + b_{10,3} - (K_3 - 1) \cdot y_{10,3} \geq 0 \\
& y_{10,1} + y_{10,2} + y_{10,3} \geq 1
\end{aligned}
$$

$$
\begin{array}{lll}
C_{11}: & b_{5,1} - K_1 \cdot y_{11,1} & \geq 0 \\
& b_{4,2} + b_{10,2} - (K_2 - 1) \cdot y_{11,2} & \geq 0 \\
& b_{3,3} - K_3 \cdot y_{11,3} & \geq 0 \\
& y_{11,1} + y_{11,2} + y_{11,3} & \geq 1 \\
C_{12}: & b_{4,2} + b_{5,2} - (K_2 - 1) \cdot y_{12,2} & \geq 0 \\
& b_{3,3} + b_{8,3} + b_{10,3} - (K_3 - 2) \cdot y_{12,3} & \geq 0 \\
& y_{12,2} + y_{12,3} & \geq 1 \\
C_{13}: & b_{3,2} + b_{4,2} + b_{5,2} + b_{9,2} - (K_2 - 3) \cdot y_{13,2} & \geq 0 \\
& b_{8,3} + b_{10,3} - (K_3 - 1) \cdot y_{13,3} & \geq 0 \\
& y_{13,2} + y_{13,3} & \geq 1 \\
C_{14}: & b_{4,2} + b_{5,2} + b_{10,2} - (K_2 - 2) \cdot y_{14,2} & \geq 0 \\
& b_{3,3} - K_3 \cdot y_{14,3} & \geq 0 \\
& y_{14,2} + y_{14,3} & \geq 1 \\
C_{15}: & b_{10,1} - K_1 \cdot y_{15,1} & \geq 0 \\
& b_{4,2} - K_2 \cdot y_{15,2} & \geq 0 \\
& b_{3,3} - K_3 \cdot y_{15,3} & \geq 0 \\
& y_{15,1} + y_{15,2} + y_{15,3} & \geq 1 \\
C_{16}: & b_{10,1} - K_1 \cdot y_{16,1} & \geq 0 \\
& b_{4,2} + b_{9,2} + b_{3,2} - (K_2 - 2) \cdot y_{16,2} & \geq 0 \\
& y_{16,1} + y_{16,2} & \geq 1 \\
C_{17}: & b_{4,2} + b_{7,2} + b_{10,2} - (K_2 - 2) \cdot y_{17,2} & \geq 0 \\
& b_{3,3} - K_3 \cdot y_{17,3} & \geq 0 \\
& y_{17,2} + y_{17,3} & \geq 1 \\
C_{18}: & b_{2,1} + b_{4,1} + b_{5,1} - (K_1 - 2) \cdot y_{18,1} & \geq 0 \\
& b_{8,3} + b_{10,3} - (K_3 - 1) \cdot y_{18,3} & \geq 0 \\
& y_{18,1} + y_{18,3} & \geq 1 \\
C_{19}: & b_{10,2} - K_2 \cdot y_{19,2} & \geq 0 \\
& b_{5,3} + b_{8,3} - (K_3 - 1) \cdot y_{19,3} & \geq 0 \\
& y_{19,2} + y_{19,3} & \geq 1
\end{array}
$$

Note that all y_{ir} are (0,1)-variables. Table 5 shows the solution of the linear program for different values of $\underline{K} = (K_1, K_2, K_3)$. We included only variables which evaluated nonzero values. The solutions for variables b_{ir} are an optimal allocation Φ^*.

K_1	3	10	46
K_2	3	10	4
K_3	3	10	125
B^*	8	36	175
$b_{2,1}$	2	9	45
$b_{4,2}$	3	10	4
$b_{10,2}$	3	10	4
$b_{6,3}$	0	7	122

Table 5: Results for Example 2.

For $\underline{K} = (3,3,3)$ the deadlock conditions for cycle C_3 and C_{13} are satisfied without allocating any capacities (see Corollary 1). Therefore, the constraints in the linear program which were derived from cycles C_3 and C_{13} can be eliminated for the optimization.

7 Conclusions

We presented the necessary and sufficient conditions for a deadlock free blocking network with multiple routing chains. The correctness of the conditions was proved in section 3. Then, we addressed the problem of finding an optimal deadlock free buffer allocation for a given network. We presented a general algorithm which generates an optimal allocation for multiple chain networks without any restriction on the topology. The algorithm was made efficient by an improved method to find a certain subset of cycles in a directed graph. This method was discussed and evaluated in section 5. Examples in section 6 demonstrated the application of the optimization algorithm.

References

[1] I. F. Akyildiz and H. G. Perros, "Queueing Networks with Finite Capacity Queues", *Special Issue in Performance Evaluation*, Vol. 10, No. 4, December 1989.

[2] S. Kundu and I. F. Akyildiz, "Deadlock Free Buffer Allocation in Closed Queueing Networks", *Queueing Systems*, Vol. 4, No. 1, January 1989, pp. 47 - 56.

[3] I. Yilmazyildiz and Y. Onozato, "Deadlock Free Conditions for the Buffer Capacities in Multiple Class Queueing Networks", *Manuscript*, University of Electro-Communications, Tokyo, Japan, March 1989.

[4] D. B. Johnson, "Finding all the Elementary Circuits in a Directed Graph", *SIAM Journal on Computing*, Vol. 4, No. 1, March 1975, pp. 77 - 84.

[5] L. Schrage, "Linear, Integer, and Quadratic Programming with LINDO", *The Scientific Press*, Third Edition, 1986.

Appendix

The following algorithm describes the extended version of *Johnson's algorithm* as discussed in section 5.1.

INPUT: $G = (V, E)$ represented by the adjacency list A_G.
OUTPUT: $\tilde{C}$
COMMENTS: A stack is used with operations POP(), PUSH(), INIT_STACK() and GET_STACK() (write current stack into a list). Set operators are used for list manipulations.

DATA STRUCTURES:

```
int                 N;
boolean             blocked[1..N];
vertex              s;
vertex list array   B[1..N];
vertex list         V_scc;
vertex list array   A_scc[1..N]
list of vertex lists Cycle;
```

```
UPDATE_CYCLE (vertex list newcycle)
  begin
    for (C ∈ Cycle) do
      if (C ⊆ newcycle)
        return;
      else if (newcycle ⊆ C)
        Cycle := Cycle \ {C};
    Cycle := Cycle ∪ {newcycle};
  end
```

```
SIMPLE_CYCLE()
  begin
    for (v ∈ V) do
      if (v ∈ A_G(v))
        begin
          Cycle := Cycle ∪ {v};
          A_G := A_G \ {A_G(v)};
          V := V \ {v};
          for (w ∈ V_G) do
            A_G(w) := A_G(w) \ {v};
        end
    for (v, w ∈ V) do
      if (v ∈ A_G(w) ∧ w ∈ A_G(v)) do
        begin
          UPDATE_CYCLE({v, w});
          A_G(w) := A_G(w) \ {v};
          A_G(v) := A_G(v) \ {w};
        end
  end
```

```
boolean CHECK_STACK(vertex list stack)
  begin
    for (C ∈ Cycle)
      if (C ⊆ GET_STACK())
        return(TRUE);
    return (FALSE);
  end
```

```
UNBLOCK(vertex n)
  begin
    vertex w;
    blocked[u] := FALSE;
    for (w ∈ B[u]) do
      begin
        B[u] := B[u] \ {w};
        if (blocked[w])
          UNBLOCK(w);
      end
  end
```

```
boolean CYCLE(vertex v)
  begin
    bool flag;
    flag := FALSE;
    PUSH(v);
    blocked[v] := TRUE;
    if (CHECK_STACK (GET_STACK()))
      flag := TRUE;
    else (w ∈ A_scc(v)) do
      for (w ∈ A_scc(v)) do
        if (w = s)
          begin
            UPDATE_CYCLE(GET_STACK());
            flag := TRUE;
          end
        else if (¬blocked[w])
          if (CYCLE(w))
            flag := TRUE;
    if (flag)
      UNBLOCK(v);
    else
      for (w ∈ A_scc(v)) do
        if (v ∉ B[w])
          B[w] := B[w] ∪ {w};
    POP();
    return(flag);
  end
```

```
begin
    Cycle := ∅;
    SIMPLE_CYCLE();
    CLEAR_STACK();
    for (v ∈ V) do
        begin
            A_scc := adjacency structure of SCC with
                     least vertex in subgraph of G induced by {n, n + 1, ..., N};
            V_scc := vertex list corresponding to A_scc;
            if (|V_scc| > 2)
                begin
                    s := vertex with least index in V_scc;
                    for (i ∈ V_scc) do
                        begin
                            blocked[i] := FALSE;
                            B[i] := ∅;
                        end
                    CYCLE(s);
                end
        end
    for (C ∈ Cycle) do
        PRINT(C);
end
```